FAST COMPRESSION OF IMAGERY WITH HIGH FREQUENCY CONTENT

THESIS

D. Scott Anderson

AFIT/GE/ENG/03-21

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/03-21

FAST COMPRESSION OF IMAGERY WITH HIGH FREQUENCY CONTENT

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering
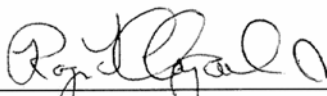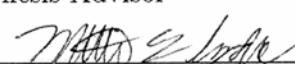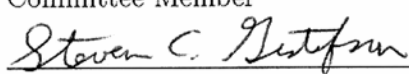
D. Scott Anderson, B.S.E.E., University of Dayton

March, 2003

AFIT/GE/ENG/03-21

FAST COMPRESSION OF IMAGERY WITH HIGH FREQUENCY CONTENT

D. Scott Anderson, B.S.E.E., University of Dayton

Approved:

<table>
<tr><td>_____</td><td>10 MAR 03</td></tr>
<tr><td>Maj Roger L. Claypoole, Jr., Ph.D.
Thesis Advisor</td><td>Date</td></tr>
<tr><td>_____</td><td>10 MAR 03</td></tr>
<tr><td>Maj Matthew E. Goda, Ph.D.
Committee Member</td><td>Date</td></tr>
<tr><td>_____</td><td>10 Mar 03</td></tr>
<tr><td>Dr. Steven C. Gustafson, Ph.D.
Committee Member</td><td>Date</td></tr>
</table>

Acknowledgements

At this point, I wish to extend my gratitude to the committee members who reviewed this thesis: to Dr. Steven Gustafson whose intelligent review of this research greatly improved the end product, to Maj. Matthew Goda who offered some wonderful ideas supporting my research goals, and especially to Maj. Roger Claypoole, Jr. who extends himself to his students in ways that deeply impact them academically and professionally. I would also like to thank the sponsor, Dr. Jack Parker, for supporting this research, providing invaluable academic advise and career opportunities. Lastly, I thank my mentor, Dr. Mark DeLong, for teaching me many of the professional lessons that can't be learned in a classroom.

D. Scott Anderson

Table of Contents

List of Figures

List of Tables

AFIT/GE/ENG/03-21

Abstract

Image compression is an active research area due to the many applications involving electronic media. Much research has been focused on image quality versus bit rate and/or algorithm speed. Here, we seek an effective image coder with a weighted constraint on speed. However, the compression must not taint the quality of impulsive features in the image. Moreover, the camera is operated in a mode that creates a dominant fixed pattern noise across the image array, degrading visual quality and disrupting compression performance. We propose a method that efficiently compresses such an image. We begin by characterizing and removing the fixed pattern noise from the image, thereby dramatically improving its visual quality. We follow noise removal with a histogram, or contrast, stretch. We then choose a transform that can be implemented rapidly with basic arithmetic operations and suggest a fast way to code the transform coefficients.

Image quality is a great concern. We are particularly interested in preserving pixel-sized impulsive features in the image. We seek a method of quantifying image quality based not only on the high energy low-frequency objects but also on the smaller energy, high-frequency impulsive objects. Standard image quality measures, such as mean squared error (MSE), tend to emphasize the quality of high energy objects and give less weight to the quality of pixel-sized impulses. Therefore, we develop a new measure that gives high-frequency impulsive features a greater contribution than MSE to the overall quality.

x

FAST COMPRESSION OF IMAGERY WITH HIGH FREQUENCY CONTENT

I. Introduction

1.1 Problem Statement

We have an image containing high-frequency, impulsive phenomena, taken on an array that contains a specific noise pattern. We desire a transform image coder that effectively and efficiently compresses the image as quickly as possible while maintaining the quality of impulsive features in the image.

Such a coder requires several steps. In the first step, the noise pattern, which is fixed, must be characterized and removed from the image. This step is important for two reasons. One, the noise pattern severely degrades visual content in the image, making it difficult to visually discriminate objects. Two, the noise pattern degrades compression performance. It is important that we remove the pattern before compression because lossy image compression changes the character of the noise, degrading our ability to remove it afterwards.

In the second step, we must design an image coder with two main criteria. First, it must be fast. This criterion drives many important decisions in the design, such as what sort of transform we choose, how we implement the transform, how we code the coefficient information, and how we optimize the compression performance. Second, the image coder must not degrade the quality of impulsive objects in the image. The compression scheme is useless for our application if impulsive features are not preserved.

Once we finalize the coder design, we seek to measure the performance of the coder. The three main measures that we would like to quantify are speed, bit rate

(or final file size), and resulting image quality. Quantifying speed is not necessarily a trivial matter because there are many factors that come into play, such as computer clock rate, compiler performance, code optimization, and hardware configuration. Moreover, it might not be helpful to quantify time without some sort of comparison to a standard compression scheme with the same implementation configurations. Measuring bit rate is more straight forward, since arriving at this measure is only a matter of measuring the final file sizes. Quantifying image quality is, perhaps, the most nontrivial process. Image quality is relative; what looks good to a human might not be acceptable for an algorithm and vice versa. As we wish to build an image coder that maintains impulsive features, it is desirable to have a measure that reflects the quality of high-frequency phenomena. Industry standard quality measures, such as mean square error (MSE) and peak signal to noise ratio (PSNR, which is based on MSE) tend to emphasize the quality of low frequency phenomena. The problem statement is, then, extended to the design of a quality measure that gives impulsive features a greater contribution to image quality.

1.2 Thesis Organization

In chapter 2, we explain the background material on which this research is based. Chapter 3 is devoted to the methodology and begins by explaining how we characterize and remove the fixed pattern noise on the CCD array. We then design a fast transform that quickly decorrelates the image data. After analyzing the statistics of the transform coefficients for three test images, we design quantization levels for the coefficients. We then discuss how we code the image in a timely way. Finally, we suggest a new image quality measure that gives impulsive features more weight in the measure than the traditional MSE and PSNR measures. Chapter 4 contains the results of our experiments. In this chapter, we compare our image coder, designed with our specific criteria, to the JPEG algorithm in terms of implementation time,

bit rate, and image quality using PSNR and our new measure. Chapter 5 concludes with a review of the thesis and recommendations for further research.

II. Background

2.1   Introduction

There are two fundamental types of image coders for compression. The first is a spatially-based encoder, which quantizes and encodes the pixel values. The second type is the transform image coder. The image compression routine outlined in this research follows the general form of a transform image coder, which has three main parts, as shown in Figure 2.1 [9]. The first part is the transform. The transform decomposes the image into a set of coefficient-weighted functions. The second step is quantization. Here the coefficients from the transform are projected into a finite set of elements, or quanta. The third step is coding.

This chapter discusses the transform image coder and the specific configurations that are used for this research. It begins with a general discussion of transforms and why they are good for image compression. We then discuss different quantization tools and go over strategies for assigning quanta to the transform coefficients. The discussion continues with the concept of entropy coding. Finally, we discuss current industry-standard compression measures used to quantify compression algorithm performance.
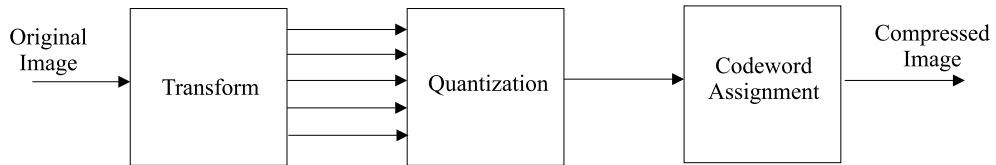
Figure 2.1.   *This block diagram represents the main steps of transform coding. The transform step converts the image into a series of transform coefficients, such as frequency or wavelet coefficients. The quantization step breaks the coefficient dynamic range into a finite number of magnitude levels. The coding step codes each quantization level with a unique identifier.*

2.2   Transforms

Transforms are extremely useful for image compression because they reduce correlation, or redundancy, in an image [9]. In the spatial domain of image scenery, there is typically a large amount of redundant information; each object in the image is usually represented by several pixels that are similar in relative value or color. A transform ideally removes the redundancies so that important image information is coded only once.

Besides reduced pixel correlation, another desirable property in transforms is energy compaction [9] [1]. A transform with good energy compaction will put most of the energy from the image in just a fraction of the transform coefficients. In this form, only a few dominant coefficients describe the image. The simplest example can be described with the Fourier Transform of a flood filled image. In the spatial domain of a flood filled image, every pixel has the same value. However, in the frequency domain, all pixel energy is concentrated in just one data element: the DC or zero-frequency coefficient. So the energy compaction property of the Fourier Transform allows us to express all the energy of our simple flood filled image with just one coefficient instead of all of the spatial domain pixels of the image.

2.2.1   The Wavelet Transform.   Our image coder is based on the discrete wavelet transform. Wavelets have been used in mathematics, and, more recently, in signal processing, as a way to analyze data components localized in both time (or in the case of imagery, space) and frequency [3] [1] [17]. In discrete wavelet analysis, information is decomposed into a series of scaled high-frequency detail coefficients and low-frequency coarse coefficients. A filter bank representation of a simple DWT is shown in Figure 2.2. In the figure, $c_p(n)$ and $d_p(n)$ are the coarse and detail coefficients, respectively, at the pth scale. We iterate on the coarse coefficients creating a set of more-coarse coefficients and a set of band-limited detail coefficients at the next frequency scale. The filters $h(z)$ and $g(z)$ are the wavelet filters and are

typically low pass and high pass filters, respectively. They are not the actual scaling and wavelet functions. However, they dictate properties that govern the validity of the wavelet transform (see [3], [1], and [17]).
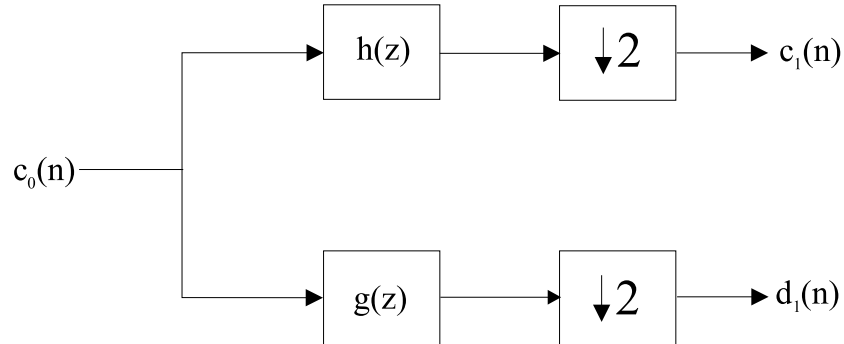
Figure 2.2.　*The block diagram in this figure is a general implementation of the discrete wavelet transform. Here the scale and wavelet filters are applied before the downsampling operation to create the coarse and detail coefficients, respectively.*

　　　2.2.2　The Lifting Implementation.　In the mid 1990's, the lifting scheme was systematically developed to provide a simple way to calculate the discrete wavelet transform [16] [2] [1]. The first step in lifting, as shown in Figure 2.3, is to split the data into even and odd data elements. We use the even set to predict the odd set using a prediction filter. The difference between the odd set and our prediction of the odd set based on the even set makes up the detail coefficients. The coarse coefficients are created when we update the even set of data elements with the newly created detail coefficients using an anti-aliasing, or update, filter. The last step is the normalization step, which may be needed to make a valid wavelet transform. Again, we iterate on the coarse coefficients to create different scales of the lifting transform.

　　　This space-domain construction of the discrete wavelet transform has several advantages [16] [1]. First, the ladder construction makes the transform fast and easy to compute. Second, a rounding step can be added at the output of each filter in the lift to create an integer to integer transform, which can increase calculation speed.
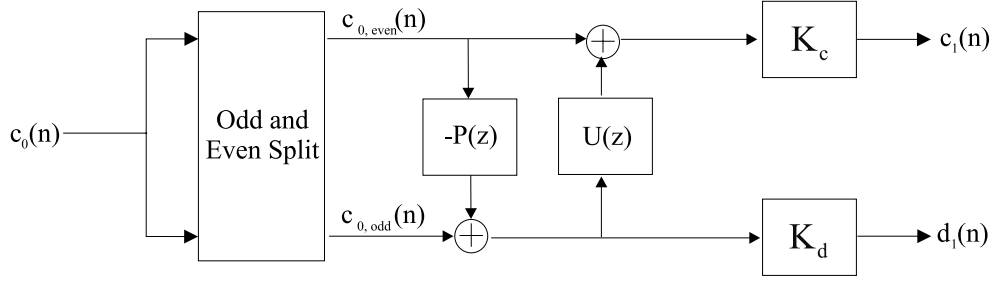
Figure 2.3.   *This block diagram of the lifting implementation demonstrates some of the advantages of using the lift to calculate the DWT of an image, such as the ladder-like construction.*

Third, each iteration of the lift is easily invertible. Fourth, the lift is implemented with in-place calculations, meaning no further memory is needed to compute the transform. Last, any wavelet transform can be implemented with a lift, although multiple steps might be needed.

2.3   Coefficient Quantization

In the quantization step, we project the coefficients into a finite set of elements, called quantization levels or quanta [9]. This projection is a non-linear transformation, an example of which is shown in Figure 2.4. Note that the result of the transformation $Q(x)$ is merely a discrete representation of x. If x is already discrete, then $Q(x)$ is a discrete representation of x with fewer quanta. Quantizing induces quantization error. The mean-square quantization error, which is based on the probability density function (pdf) of x, is

$$\epsilon_q = \int_{-\infty}^{\infty} f_x(x)\,(x - Q(x))^2\,dx \tag{2.1}$$

where $f_x(x)$ is the pdf of x. In compression there is a tradeoff between the number of quantization levels and ϵ_q. If we decrease the number of quantization levels, we decrease the number of symbols needed to represent $Q(x)$, which is good for compression, but we increase ϵ_q, which is bad for image quality. Current industry

2-4

practice often uses the Lloyd-Max quantization scheme to determine the quantization values because it optimally minimizes quantization error in the mean square for a given number of quantization levels [10] [11].
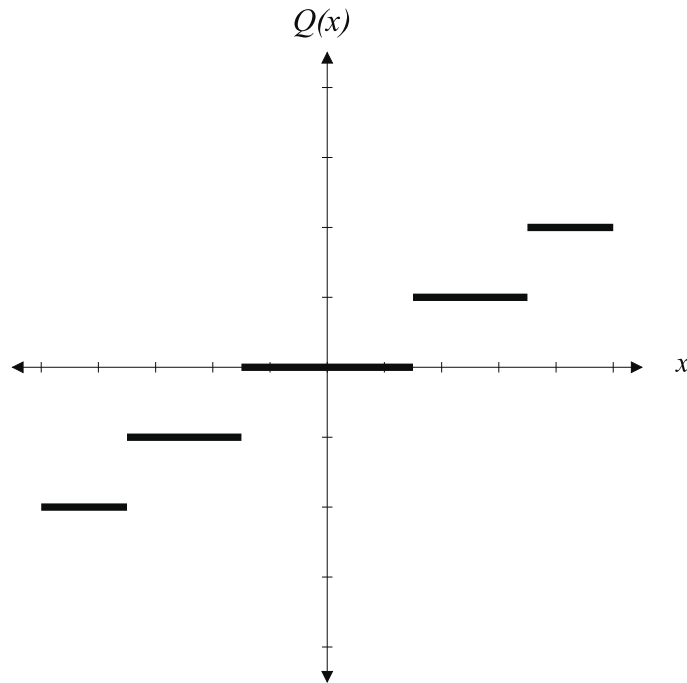
Figure 2.4.   *In the quantization process, the coefficients are projected into a finite set of elements, called quantization levels. These levels are discrete representations of the original coefficients. A simple example of a quantizer is shown in this graph. Here, the output of the quantizer is $Q(x)$.*

2.3.1 Thresholding.   Thresholding is a form of quantization. However, the goals of thresholding are somewhat different than that of quantization. In thresholding transform coefficients, we discard coefficients that we deem insignificant. Because most of the energy in the image is contained in only a fraction of the transform coefficients due to the energy compaction property, many of the "insignificant" coefficients can be discarded with minimal effect on overall image quality.

There are two main types of thresholds: hard-thresholds and soft-thresholds. In a hard-threshold, the values below the threshold are simply set to zero. In a

soft-threshold, the values below the threshold are set to zero and all other values are shifted toward zero by the threshold value. A soft-threshold transformation is shown in Figure 2.5.
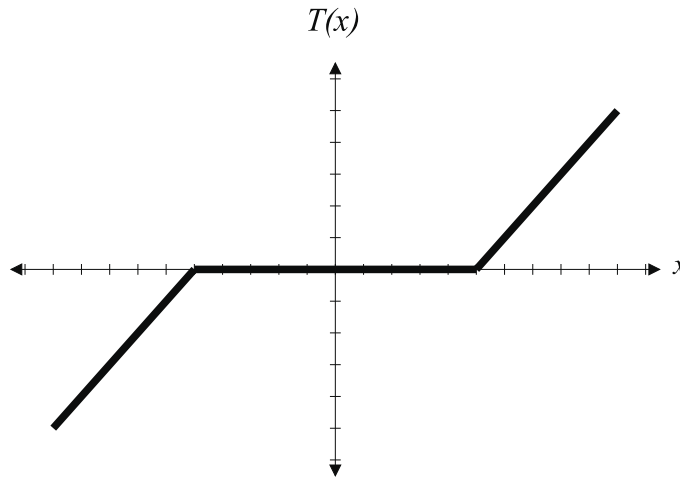
Figure 2.5.   *This graph represents a soft threshold. The values within the range of the threshold are set to zero and the other values are shifted towards zero by the threshold amount.*

    2.3.1.1   Denoising.   It should be noted that thresholding wavelet detail coefficients provides excellent denoising properties [4]. As explained in Section 2.2, the wavelet transform decorrelates image pixels and puts a majority of the image information in just a few of the coefficients. However, noise, if white, is uncorrelated, so the decorrelating properties of the transform have no effect. Therefore, in the transform domain, the important image information, which is parsimoniously stored in a just a few of the wavelet coefficients, is much stronger than and usually separable from the noise information. A hard-threshold of the weaker wavelet coefficients will affect the noise much more than the image information, resulting in a reconstructed image with drastically reduced noise.

2.4 Entropy Coding

Entropy, in information theory, is the theoretical minimum amount of information needed to represent a series of realizations of a discrete random variable. In our case, the term "amount of information" takes the form of average number of bits. Once we estimate the probability mass function of our quantized levels, we can design Huffman codes that code the information nearly to entropy [6].

2.5 Image Compression Measures

In image compression, the main trade off for bit rate is image quality. For this reason, compression performance is quantified by a plot of image quality versus bit rate. Bit rate is usually measured in bits per pixel (bpp). It is not uncommon, and is usually more meaningful, to refer to compression ratio instead. Compression ratio is the ratio of the uncompressed file size to compressed file size.

What quantifies the quality of an image is somewhat more relative than the simple matter of measuring file sizes. Image quality depends on the application; different applications may consider certain features more important than others. For example, in our application, we consider single pixel phenomena to be extremely important. In many other applications, the quality of a single pixel is negligible compared to the rest of the image.

One of the most popular metrics used to measure the visual quality of an image is the peak signal to noise ratio (PSNR) [12]. PSNR measures the average pixel fidelity between the original and compressed image and is

$$PSNR(I_c, I_o) = 10 \log_{10} \left(\frac{\max(I_o)^2}{MSE(I_c, I_o)} \right) \tag{2.2}$$

where I_c is the compressed image, I_o is the original image, and $\max(I_o)$ is the maximum possible value of I_o, which, for 8-bit data, is 255. $MSE(I_c, I_o)$ is the mean squared error between the original and compressed image and is

$$MSE(I_c, I_o) = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} (I_o(i,j) - I_c(i,j))^2. \qquad (2.3)$$

With PSNR, a higher measure means higher image quality. With MSE, a lower measure means higher image quality.

2.6   Conclusion

In this chapter, we began by discussing the elements of the transform image coder. There are three main steps in such a coder: the transform, the quantization, and the codeword assignments. The transform is a process that converts spatial image information to a form that is usually very different than its original state. With a transform, we seek to reduce redundancies in the image so that important image information is coded only once. Transforms also provide energy compaction, meaning that most of the energy in the image is stored in only a fraction of the transform coefficients, allowing us to discern which coefficients are insignificant. After briefly discussing the generic elements of the transform image coder, we discussed the discrete wavelet transform and the lifting implementation of the transform. We continued with the idea of quantization and referenced the Lloyd-Max scheme of calculating the quantization levels that optimally minimize the quantization error in the mean square. Our discussion then turned to codeword assignments. If we know the probability mass function of the quantization levels, we can produce a set of Huffman codes that code the data nearly to entropy. We concluded with a brief discussion of industry-standard image-compression measures that are commonly used to describe the performance of an image coder.

III. Methodology

3.1 Introduction

In this chapter, we discuss the method for achieving our research goals. To review, the goals of this research are to design a scheme that effectively compresses an image with weighted constraints on algorithm speed and the preservation of high frequency detail in the image. The chapter begins with an analysis of the image. Due to application constraints, the camera must be run in a mode that creates a dominant noise pattern across the image array. This noise pattern is fixed and repeatable on the array and, if characterizable, should be removed because it clearly degrades the viewable information in the image. It is also important to remove the pattern *before* compression because lossy compression will change the character of the noise and degrade our ability to remove it afterwards. Once we remove this fixed pattern noise from the image, we show the necessity of stretching the image histogram to maximize compression performance. This histogram stretch does not cost us anything in terms of processing speed because it can be implemented at the same time as the fixed pattern noise correction.

Next we discuss the lifting implementation of the wavelet transform as it applies to our routine. During this discussion we choose a transform based on speed and simplicity. We also remove the normalization factor from the lifting implementation for the sake of computational simplicity. Although this step invalidates the process as a wavelet transform, we still preserve many of the desirable wavelet properties. The discussion then turns to a statistical analysis of the transform coefficients. The detail coefficients of the transform are used to determine proper quanta that minimize quantization error in the mean square using the Lloyd-Max quantization scheme. A brief discussion follows describing how a soft-threshold on the detail coefficients has minimal impact on the optimality of the Lloyd-Max quantization values, due mainly

to the fact that the detail coefficients of the transform tend to follow a Laplacian distribution.

Once the transform coefficients are properly quantized, the simple matter of how to store the coefficients remains. Both the values of the quantized coefficients and their locations are stored. Due to thresholding, many of the coefficients are determined to be insignificant and are designated as zero. In order to gain compression, we choose a run-length coding scheme to store significant coefficient locations by simply counting the number of zeros between each one. The number of consecutive zeros between each significant coefficient is decomposed into a combination of numbers that are assigned codewords.

Finally, we introduce a new measure of image quality that is specifically designed to measure the fidelity of pixel sized objects in our compressed image. It is shown that current image quality measures, like PSNR or MSE, tend to emphasize fidelity to low frequency content in an image. To measure the performance of our image compression routine, we choose to consider every frequency component of the image as equally important. This choice makes sense considering that the type of objects that we want to maintain in our compressed image, single impulsive objects, have infinite frequency content.

Before we begin, we introduce three test images that we use for the research. These images, shown in Figures 3.1-3.3, clearly show the dominant fixed pattern noise. These images were chosen because they are representative of the type of imagery that we wish to compress. Test image 1, in Figure 3.1, was chosen because it was sunny when we took the image and there are single-pixel sized impulsive features in the scene. Test image 2, Figure 3.2, is a typical scene on a darker-overcast day. Test image 3, Figure 3.3, is a typical scene on a lighter-overcast day. This image also contains larger impulsive features.

Figure 3.1. *Test Image 1, which is the first test image used for the research. This image was chosen for two main reasons: The scene was taken on a sunny day and there are single-pixel sized impulsive features in the scene.*
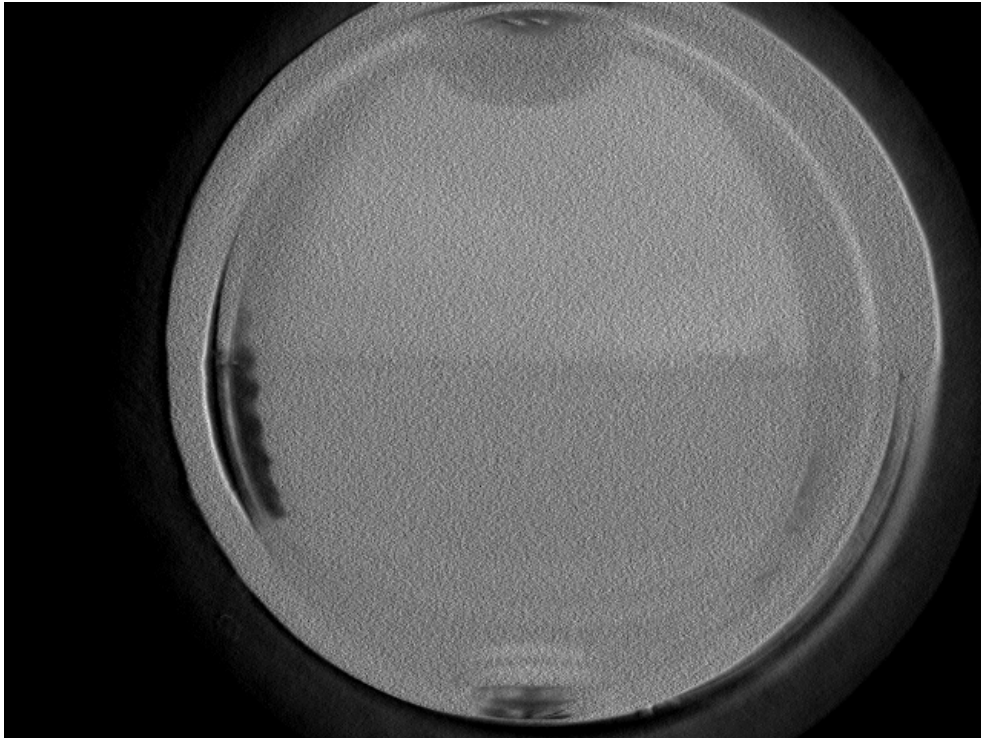
Figure 3.2.   *Test Image 2. This test image is typical for imagery taken on a dark-overcast day.*

Figure 3.3. *Test Image 3. This test image is typical for imagery taken on a lighter-overcast day. There are also large impulsive objects in the scene.*

3.2 Characterizing the Fixed Pattern Noise on the CCD

The CCD camera produces images that contain a very strong fixed pattern noise (FPN) across the whole array. This fixed pattern noise is caused by synchronous timing generation effects from fast data rates [5] and from on-board DSP anomalies. There is a setting on the camera that attempts to mitigate the effect of this noise pattern by slightly blurring the image. However, our application requires that we maintain fidelity to single pixel sized objects in the image, so slightly blurring the image to reduce the fixed pattern noise is not an option. Instead, we choose to characterize the FPN and remove it from the image.

The FPN likely takes the form of a different bias and responsivity factor for each pixel. We begin our experiment by assuming that each pixel of the fixed noise pattern can be normalized to the CCD array mean by a first order polynomial transformation. The responsivity portion of this model represents the multiplicative portion of the characterization. The DC bias represents the additive portion of the characterization. If we were to plot out each pixel-element value from the noise pattern against the intensity of the incoming light using this model, we would expect to see a slightly different straight line for every pixel. However, our initial assumptions in this matter are not correct; the characterization curve is nonlinear and can not be represented by only one polynomial.

3.2.1 Setting up the Measurements. The characterization measurement is set up by placing the camera with the focusing lens removed so that it is staring directly into the exit port of an optical integration sphere, as shown in Figure 3.4. A tungsten lamp is placed to illuminate the entrance port of the integration sphere, and an iris controls the throughput of light into the sphere from the entrance port. The result is an image where every CCD pixel is exposed to the same amount of light. This type of image is called a flood fill image. Throughout the experiment, the luminescence of the tungsten lamp was kept constant with the aid of a direct-

feedback light detector. The light intensity exposed to the CCD is controlled by the opening and closing of the iris.
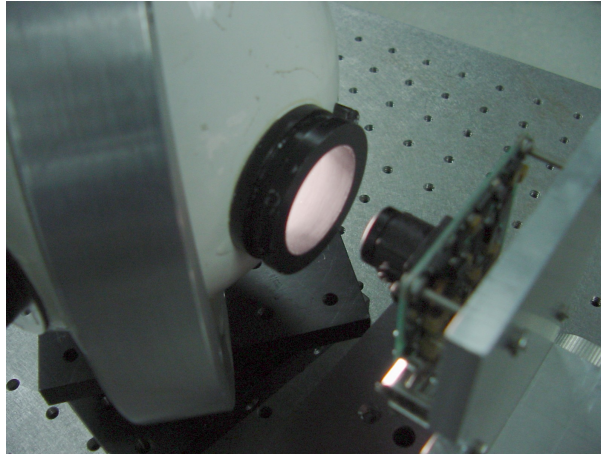
Figure 3.4.   *This picture demonstrates the position of the camera with respect to the exit port of the integration sphere during the fixed pattern noise characterization measurements. Note that during the actual measurements, the camera aperture is inserted directly into the exit port. A tungsten lamp illuminates the interior of the sphere from the entrance port and an iris is used to control the light intensity.*

3.2.2   Taking the Measurements.   We chose to collect imagery with seven different light intensity levels for the characterization measurements. For each intensity setting, we collect forty images and average them to reduce the effect of thermal noise on the measurements. The lowest light intensity measured was with the lamp turned off for an average pixel value of zero. The highest light intensity created an average pixel value of about 200.

Based on these seven measurements, we discovered two distinct characterization regions for each pixel. The discontinuity is clearly seen from the overlay of the response of 10 different pixels in the plot in the Figure 3.5. In the first region, where the average pixel intensity is below 100, the characterization seems to contain very small bias components. Above the average intensity value of 100, the bias component becomes the dominant feature in the characterization.

The difference between the two regions are also clearly seen in Figure 3.6, which shows a plot of the seven intensity values for several adjacent pixels in an image line. Note that above a certain intensity threshold (about 100), an additive noise pattern seems to take affect that is not present for lower intensity levels. The fact that there are some data points in the fourth brightest intensity measurement that are lower in value than those same elements in the third brightest measurement supports the idea of having two characterization regimes. Although it is common in CCD cameras to contain a fixed pattern noise, having two distinct patterns as a function of intensity is not common. The reason that our camera has these two distinct characterization regions is, according to the camera vendor, an unexplainable on-board DSP anomaly.
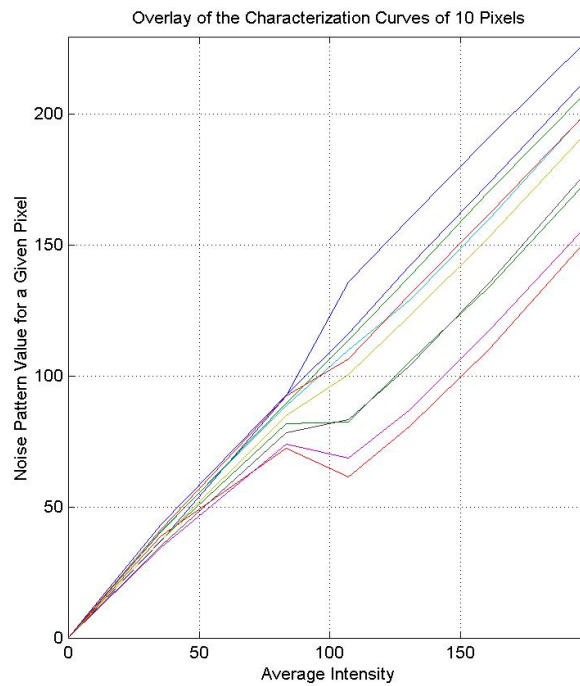
Figure 3.5.   *This graph shows the fixed pattern noise effects. It shows the intensity of 10 pixels versus the average pixel intensity on the array. The discontinuity of the characterization is very clear.*

3.2.3   Applying the Characterization to Remove the Pattern Noise.   The fact that there are two distinct characterization regions gives rise to the question of
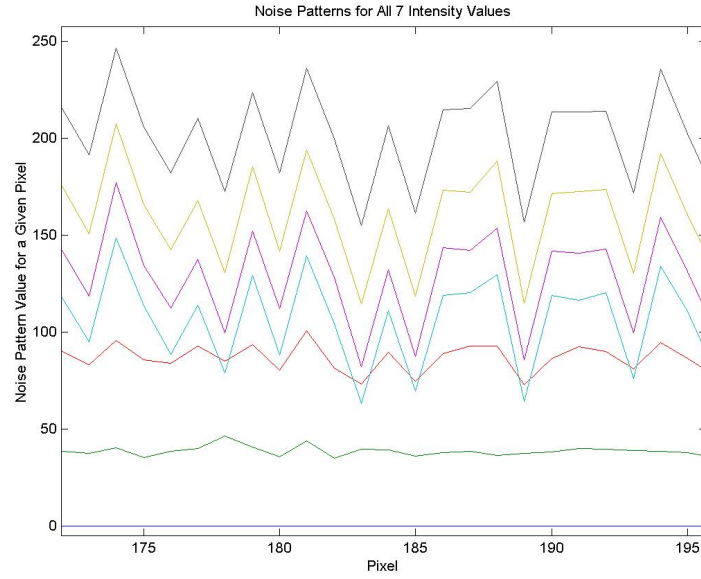
Figure 3.6.   *This plot shows the values of several adjacent pixels for seven different light intensities.*

which region to apply to which pixels. It is not valid to simply test to see if each pixel is below some threshold in order to determine which region to apply because, as demonstrated in Section 3.2.2, some of the pixels can have values that could map to both regions. It is possible to base the decision of which characterization region to apply on a local average, but that would add complexity and implementation time to the algorithm. As we will discuss in Section 3.3, the information of interest in this imagery is consistently contained within a section of dynamic range entirely within the more light-intense region. Therefore, we apply the second correction region to all pixels.

Our goal is to normalize each pixel separately to the average pixel value on the array. Based on the data we collected, it seems reasonable to consider that each pixel has a slightly different multiplicative response to light and a slightly different additive bias. Assuming this model, we can induce a fixed pattern noise on a flood fill image by simply multiplying each pixel's responsivity factor by some constant

that represents the mean pixel value in the array and then adding the bias at each pixel. Mathematically, this is

$$IM_{FNP}(i,j) = C_{avg} \times m_{FPN}(i,j) + b_{FPN}(i,j) \tag{3.1}$$

where $IM_{FPN}(i,j)$ is the flood fill image, C_{avg} is the average pixel value of the flood filled image, $m_{FPN}(i,j)$ is the responsivity (multiplicative) portion, and $b_{FPN}(i,j)$ is the additive bias. Using the method of least squares, we employ the collected flood-fill imagery to find the values of m_{FPN} and b_{FPN} for each pixel that minimize the mean square error. By rewriting Equation 3.1, the transform that removes the FPN becomes

$$C_{avg} = \frac{IM_{FPN}(i,j) - b_{FPN}(i,j)}{m_{FPN}(i,j)}. \tag{3.2}$$

If IM_{FPN} is a regular image containing the FPN, then this simple first order transformation removes the component of the pattern based on the least square calculation of $m_{FPN}(i,j)$ and $b_{FPN}(i,j)$. Since multiplication is computationally easier to calculate than division, Equation 3.2 is easily put into the form:

$$IM_{FPNC}(i,j) = IM_{FPN}(i,j) \times \hat{m}_{FPN}(i,j) + \hat{b}_{FPN}(i,j), \tag{3.3}$$

where $IM_{FPNC}(i,j)$ is the FPN-corrected image.

As shown in Figure 3.7, the process of characterizing the noise pattern and applying the FPN correction transformation effectively removes most of the fixed pattern noise from our test image. From the figure, note that the transformation actually induced a noise pattern in the unimportant, lower intensity pixels because they are in the lower characterization region.
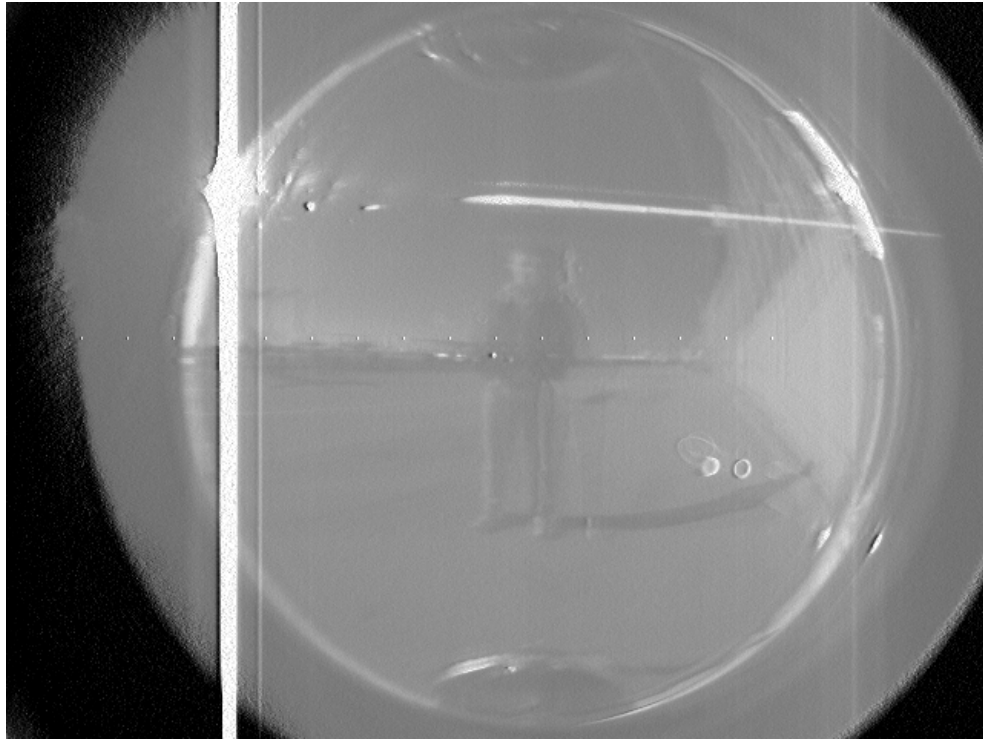
Figure 3.7.   *Test Image 1 corrected for the fixed pattern noise. After successful noise characterization, we are able to remove most of the fixed pattern noise from test image 1. Note that the transformation actually induces a noise pattern in the lower intensity, unimportant pixels because they are in the lower characterization region.*

3.3   Histogram Stretching

Figure 3.7 shows that the fixed pattern noise can be successfully removed from our image. However, the useful information in the image seems to still be compressed within a small section of the grayscale dynamic range. From a compression and a visual standpoint, it would be better to make darker the darker parts in the middle circular region, or the region that contains nearly all of the visual information, and to make lighter the lighter pixels. This process is known as histogram modification [9]. The visual benefits of histogram modification are obvious: more contrast in the middle part of the region makes it easier for us to visually discriminate objects in the image. From a compression standpoint, it is important to make the histogram modification for much the same reason. Without some sort of histogram transformation, we must invest a lot of our compression energy (in the form of quantization levels) into a region relatively shallow in dynamic range so we can discriminate objects from the compressed image. This investment leaves us limited compression energy to represent the rest of the dynamic range. What suffers is either bit rate or overall image quality. It turns out that we are able to incorporate an acceptable histogram modification into the FPN-correction process in order to implement both simultaneously. Before we discuss how, we first describe the histogram modification process.

In histogram modification, we apply a fixed or an adaptive transformation to the grayscale levels to either compress or stretch certain portions of the dynamic range. Consider the histograms of the three FPN-corrected test images in Figure 3.8. Note that the histograms are most dense between the values of 100 and 200. It is in this region, which contains nearly all of the useful visual information, that we would like to allocate more dynamic range.

There are different ways to modify the histogram to address our needs. We could choose a nonlinear transformation of the grayscale values so that values 0-100 from the histogram would be compressed to values from 0 to 10, values 100 to 200

3-12
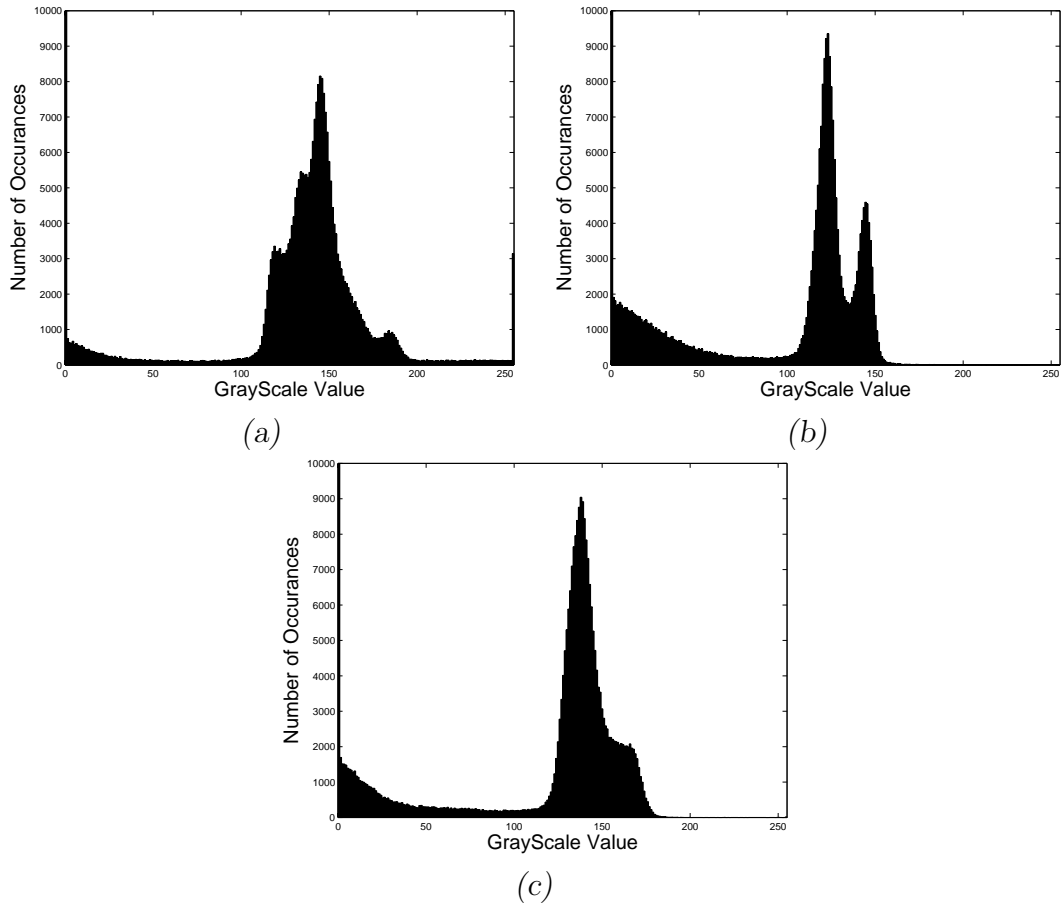
Figure 3.8.   *After the fixed pattern noise correction, our test images exhibit the histograms shown here. (a) Test Image 1 after FPN correction. (b) Test Image 2 after FPN Correction. (c) Test Image 3 after FPN correction. Note that in each image a significant portion of the histogram mass in each image is located between 100 and 200. This mass corresponds to the useful information in the image. Ideally, we want just this section of the histogram to cover the entire grayscale dynamic range.*

would be stretched to values from 10 to 245, and values greater than 200 in the histogram would be compressed into the remaining dynamic range. This approach assures us that we will not lose any information; simply remapping certain grayscales to a different value, even if the transformation makes the grayscale less significant, is better than just discarding the value. The disadvantage of this method is that the entire grayscale transformation must be done in three discontinuous parts.

Another form of histogram modification is called *histogram equalization* [9]. This method is a more automated process and seeks to evenly distribute the dynamic range to every unit of histogram mass. Such distribution means that the sections with twice as much mass have twice as much dynamic range after histogram equalization. This adaptive method is effective but calculation-intensive and takes too long for our needs. It would also place some emphasis on the very dark regions of the image which contain little visual information.

An acceptable tradeoff for speed is to discard portions of the dynamic range that we confidently believe contain negligible amounts of useable visual information. We propose to linearly stretch all of the dynamic range so that the grayscale values from 100 to 200 are remapped to the range from 0 to 255, and then clip the remaining information outside of those bounds. Figure 3.9 is a graphical representation of this transformation. Although this is not the optimal histogram modification for every image based on the histograms in Figure 3.8, this transformation will not cut out important information in the imagery and we avoid the burden of using a time-consuming adaptive histogram stretch. Another benefit for implementing this simple transformation is that it can be incorporated directly into the process that corrects the test image for the fixed pattern noise.

The implementation begins by subtracting 100 from all values and then multiplying the result by 2.55, or

$$\widehat{IM_{FPNC}}(i,j) = 2.55 \times (IM_{FPNC}(i,j) - 100), \tag{3.4}$$

3-14

Proposed Histogram Modification

Can be implemented with a single linear transformation and clipping
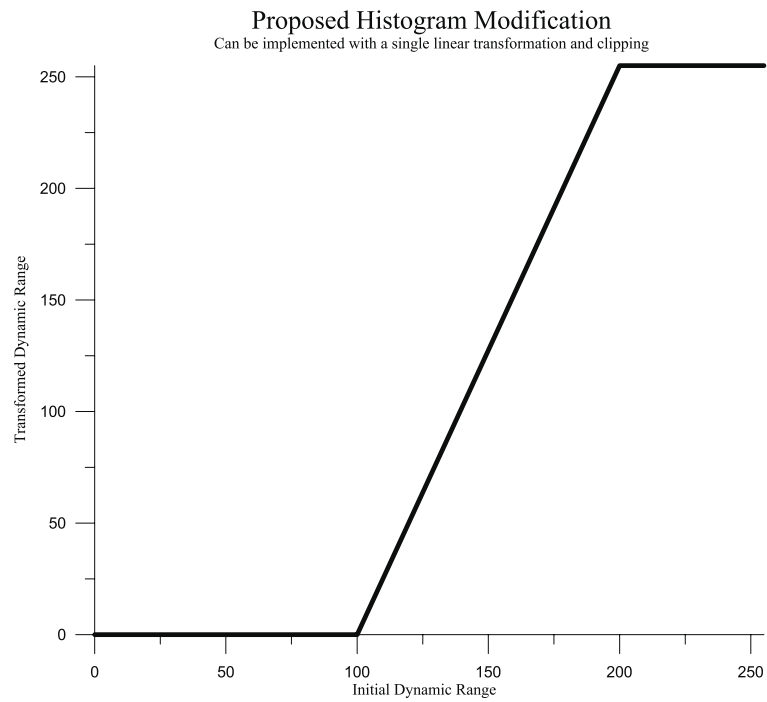
Figure 3.9.   *This transformation significantly improves the viewability and compressibility of our test image. It can be implemented with a single linear transformation and a clipping operation above and below the 8-bit dynamic range bounds.*

where $\widehat{IM_{FPNC}}(i,j)$ is the histogram-stretched version of $IM_{FPNC}(i,j)$. When Equation 3.4 is combined with Equation 3.3, we develop a new transformation that accomplishes both the fixed pattern noise correction *and* the histogram stretching. The new transformation takes the form

$$\widehat{IM_{FPNC}}(i,j) = IM_{FPN}(i,j) \times \tilde{m}_{FPN}(i,j) + \tilde{b}_{FPN}(i,j), \qquad (3.5)$$

where $\tilde{m}_{FPN}(i,j) = 2.55 \times \hat{m}_{FPN}(i,j)$ and $\tilde{b}_{FPN}(i,j) = 2.55 \times \hat{b}_{FPN}(i,j) - 255$.

After this simple transformation, we perform the following clipping operation:

$$\widehat{IM_{FPNC}}(i,j) = \begin{cases} 0 & \widehat{IM_{FPNC}}(i,j) < 0 \\ \widehat{IM_{FPNC}}(i,j) & 0 \leq \widehat{IM_{FPNC}}(i,j) \leq 255 \\ 255 & \widehat{IM_{FPNC}}(i,j) > 255 \end{cases} \qquad (3.6)$$

The histogram-stretched test imagery is found in Figures 3.10 - 3.12.

3.4 The Multi-scale Transform

After the image enhancements described in Sections 3.3 and 3.2, we design the compression process. The first step is to transform the image. We choose a wavelet-based transform so that we can take advantage of its space-localization properties for image compression. One of our goals is to create a compression algorithm that is computationally fast. The 3,1 wavelets from the Cohen-Daubechies-Feauveau (CDF) family of biorthogonal wavelets are known to be very fast because there are few filter taps and, as will be shown, the un-normalized filter taps have values of 2^{-n} and can be implemented with binary bit shifts [3] [1] [17].

As stated previously, any wavelet transform can be implemented with a lift [16] [1]. A lift is desirable for many reasons, some of which are described in Section 2.2.2. The main reason we desire the lift implementation is that it is very fast to execute. When we compute the filters needed to carry out a lift using a 1-point prediction and

Figure 3.10.   *This is test image 1 after the combined grayscale transformation of FPN removal and histogram stretch.*
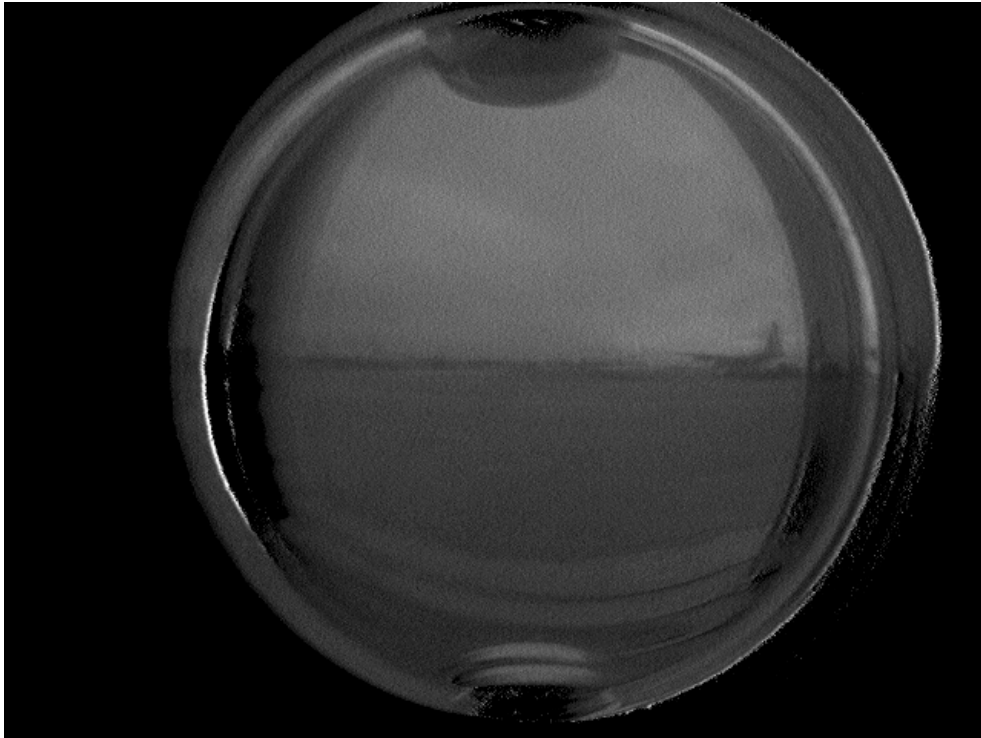
Figure 3.11.   *This is test image 2 after the combined grayscale transformation of FPN removal and histogram stretch.*
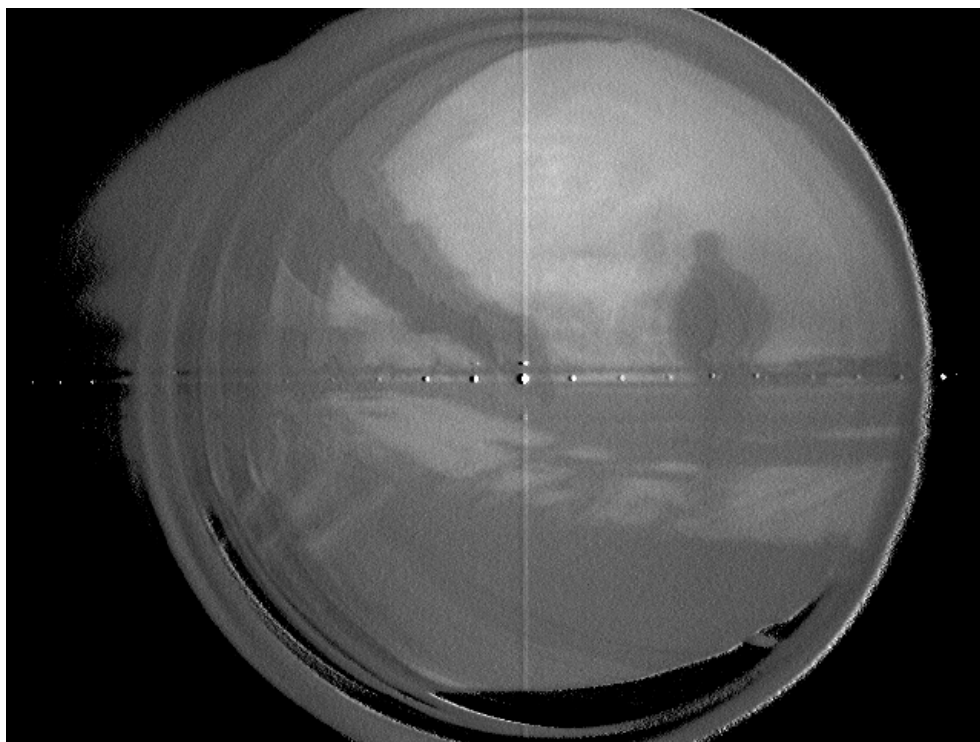
Figure 3.12.   *This is test image 3 after the combined grayscale transformation of FPN removal and histogram stretch.*

a 3-point update, as described in [2], we find that the prediction filter is $P(z) = 1$, and the update filter is $U(z) = \frac{1}{16}z + \frac{1}{2} - \frac{1}{16}z^{-1}$. These filters can be used to directly represent the unnormalized analysis and synthesis filters of the 3,1 wavelet transform. The analysis and synthesis filters, once normalized, can then be used with the wavelet recursion equations to find the actual scaling and wavelet functions on the analysis and synthesis side of the transform [3] [1] [17]. Figures 3.13 (a) and 3.13 (b) show the scaling and wavelet functions on the analysis side of the DWT, while Figures 3.13 (c) and 3.13 (d) show the scaling and wavelet functions on the synthesis side.
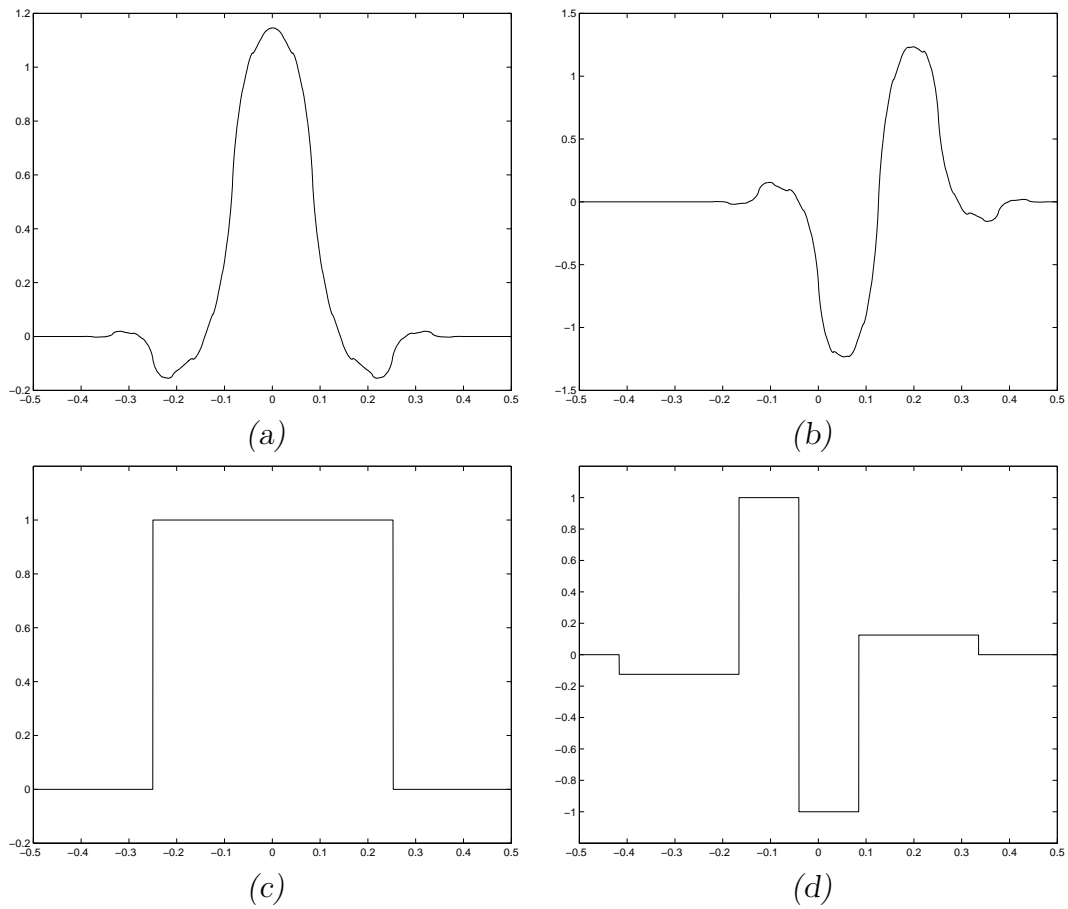
Figure 3.13.   *These are the scaling and wavelet functions of the 3,1 CDF wavelet. (a) scaling function on the analysis side. (b) wavelet function on the analysis side. (c) scaling function on the synthesis side (d) wavelet function on the synthesis side.*

The analysis side of the transform is the side that decomposes the image information into the wavelet domain. The synthesis side of the transform brings the information from the wavelet domain back into the spatial domain. It is usually advantageous in applications such as image compression to make the synthesis functions as smooth as possible so that the reconstructed image is smooth. As for the functions currently selected, the synthesis functions are not smooth and will produce blocking artifacts in the reconstructed image. However, the analysis functions are much more smooth. Therefore, we wish to swap the functions that we use for the analysis and synthesis portions of the transform. We do this by reversing the order of the prediction and update steps of the lift [2]. In reversing the order we must recalculate the filters in the reverse lift mode. First, let us discuss the normalization process in the lift.

As stated previously, there are filter properties that govern the validity of the discrete wavelet transform. Some of the properties deal with magnitude; for example, all of the filter taps in $H(z)$ must sum to $\sqrt{2}$. There are also power constraints on both $H(z)$ and $G(z)$. To make sure that these equivalent filters (whose polyphase components can be represented with the prediction and update filters) follow these properties, we must add a normalization step to the lift. Without this step the transform is not valid because the recursion equations are no longer valid.

Thus far, we have specified the wavelet transform implemented in a lift as shown in Figure 3.14. Note the reverse implementation of the lift. The corresponding wavelet filters in this implementation are $H(z) = 1 + z^{-1}$ and $G(z) = -\frac{1}{16}z^2 - \frac{1}{16}z + \frac{1}{2} - \frac{1}{2}z^{-1} + \frac{1}{16}z^{-2} + \frac{1}{16}z^{-3}$.

Taking one iteration of the DWT on an image requires that we perform the lift on each row of the image followed by the same operation on each column. The result is a data element of the same size with the four distinct quadrants of a 2-dimensional DWT. It is easy to see that when the normalization process on the coarse coefficients is $\frac{1}{\sqrt{2}}$, as it is for a valid wavelet transform, then the coarse coefficients in the LL

3-21
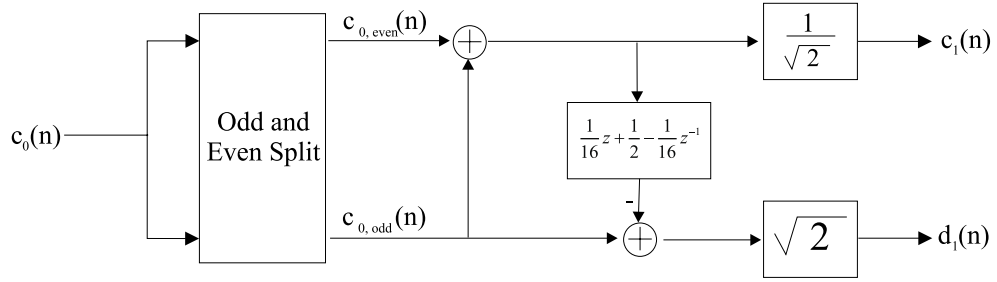
Figure 3.14.   *This block diagram shows the implementation of the 3,1 DWT implemented with a reverse lift. Note that the filters can be applied with bit shift operations.*

quadrant increase by a factor of 2 for each iteration. In our application we desire that the coefficients for every iteration have the same relative scale so that we can apply the same quantization criteria for each scale of detail coefficients. In essence, we would like the average value of the coarse coefficients at one scale to be the same as that of the coarse coefficients at the next scale. This desire is accomplished by applying a factor of $\frac{1}{2}$ to the even image data and modifying the update filter by the same factor. The prediction filter must then be multiplied by 2 in order to obtain the same detail coefficients. The normalization step is then dropped, thereby invalidating the wavelet transform. However, many desirable properties of the wavelet transform (decorrelation, energy compaction, etc.) are maintained. Figure 3.15 shows these modifications in the lift. This is the implementation that we use for our research. Note that the key elements of this lift are that it is the 3,1 DWT in the reverse lift mode, the normalization process is dropped from the end of the operation, and a factor of $\frac{1}{2}$ is added to the even data elements and carried through the filters. Since we are quantizing the result of this transform, we could, with minimal effect, round each rung on this ladder so that we implement only integer additions. Such a modification makes the transform non-linear but still easily invertible with perfect reconstruction [13].
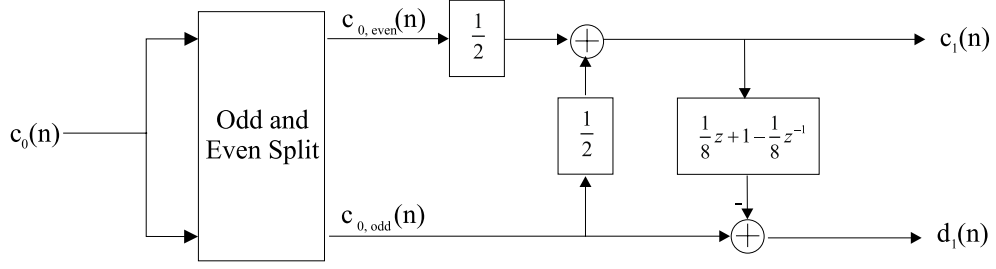
Figure 3.15. *Block diagram of the multiscale transform used for this research. Each scale of coefficients use the same relative dynamic range, the operation is done in the reverse lift mode, and the entire operation can be done with binary bit shifts and additions. Each rung on the ladder could be modified so that only integer additions are required.*

3.5 Analyzing the Transform Statistics

Assuming that a data set consists of independent realizations of some random variable, its histogram is a direct representation of the probability density function (pdf) of the random variable. The histogram shows how many times a value occurs in the data set. Thus, we can use the histogram to make assumptions about the theoretical pdf, which is used to create the Lloyd-Max quantization levels. Since we designed the transform to yield detail coefficients with the same relative dynamic range for each scale, we can combine the scales and analyze only one histogram. In Figure 3.16, the histogram of all detail coefficients for test image 1 show that the detail coefficients strongly follow a Laplacian pdf, which is

$$f_x(x) = \frac{\alpha}{2} e^{-\alpha |x|}, \tag{3.7}$$

where $E[x] = 0$, $VAR[x] = \frac{2}{\alpha^2}$ [8], and the standard deviation of x is the square root of its variance.

Although this histogram shows a strong resemblance to a Laplacian pdf, the histogram also shows that there is more probability density on the skirts of the normal Laplacian that would be better described by a Gaussian distribution. Figure 3.17 shows this effect in more detail using a logarithmic plot of the normalized

3-23
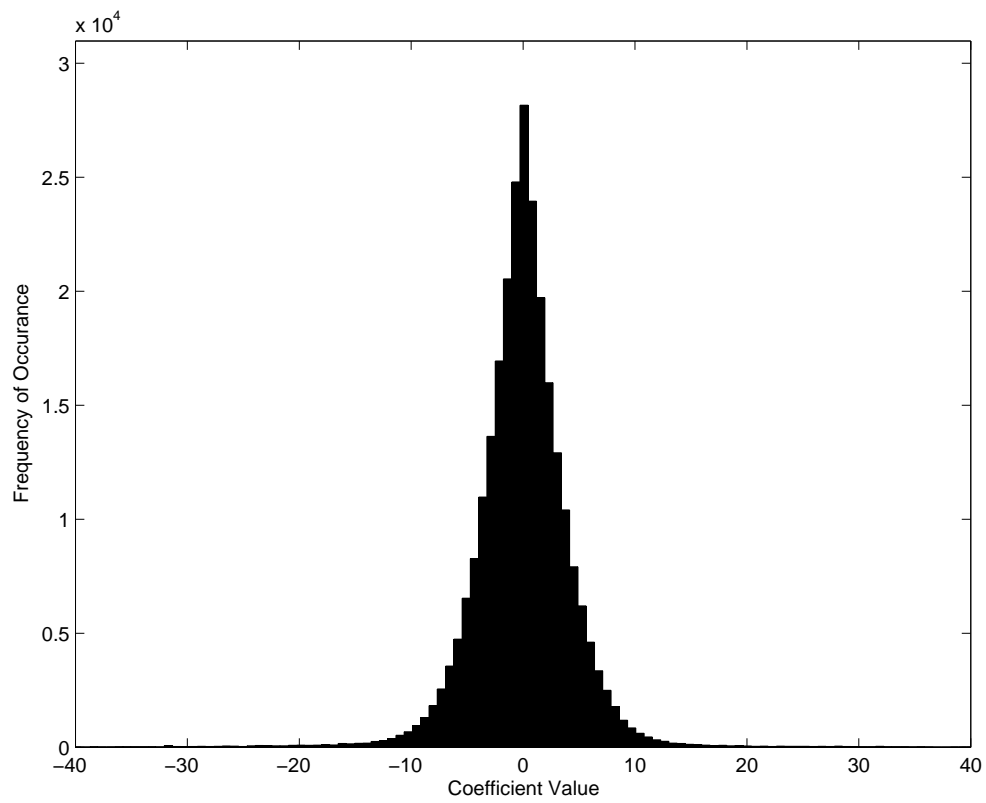
Figure 3.16.   *This is the histogram for all non-zero detail coefficients of test image 1. Note the strong resemblance of this histogram to a Laplacian pdf.*

histogram of the combined detail coefficients for all three images with an overlaid Laplacian curve. In this plot the Laplacian parameter α was empirically determined to be 0.4.
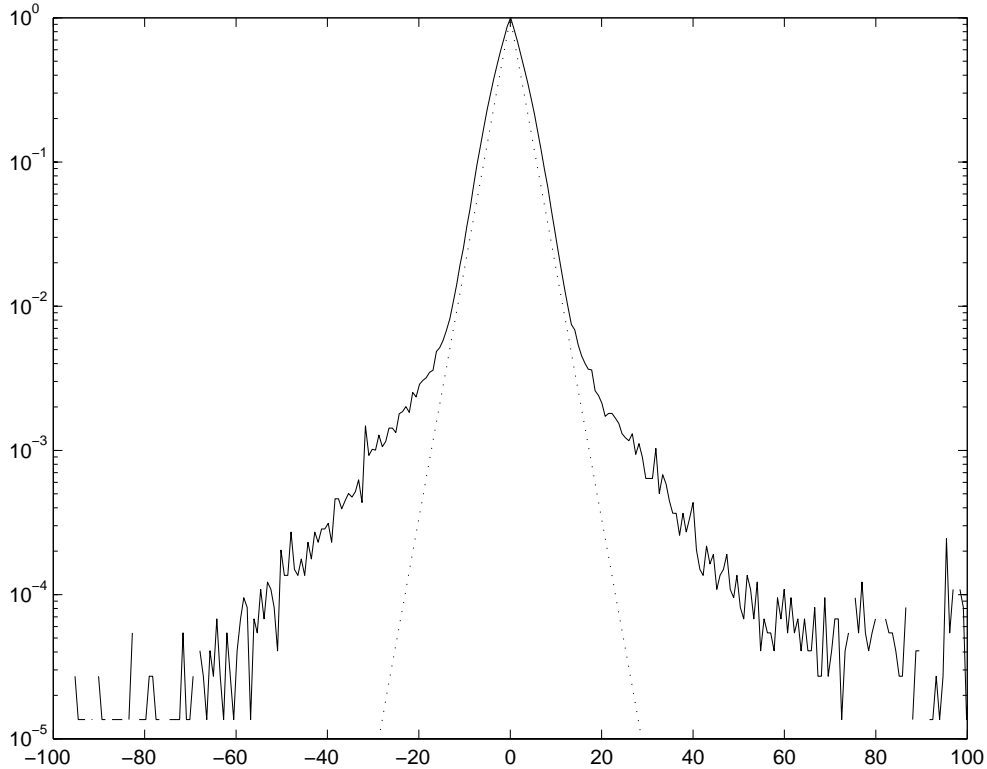
Figure 3.17.   *This logarithmic plot shows the normalized histogram of the combined detail coefficients for all scales and test images in line-form along with the normalized Laplacian-only representation. As seen from this plot, the Laplacian curve fits the histogram in the middle of the plot fairly well, but does not adequately represent the histogram mass on the outskirts of the Laplacian. In this plot, the Laplacian parameter α was empirically determined to be 0.4.*

Since there is more mass on the skirts of the histogram than can adequately be described by a Laplacian curve alone, we choose to model the histogram using a hybrid pdf made up of a Laplacian/Gaussian mixture. This new pdf has the form

$$f_x(x) = A\frac{\alpha}{2}e^{-\alpha|x|} + \frac{B}{\sqrt{2\pi\sigma_{gaus}^2}}e^{\frac{-1}{(2\sigma_{gaus}^2)}x^2},\qquad(3.8)$$

3-25

where σ_{gaus}^2 is the variance of the Gaussian pdf. For this new function to be a valid pdf,

$$A + B = 1,$$ (3.9)

where $0 \leq A, B \leq 1$.

Our empirical determination of A and B begins by counting the number of detail coefficients whose absolute value is greater than twice the standard deviation of the Laplacian curve. This number divided by the total number of detail coefficients is the empirically derived probability that the absolute value of the coefficient is greater than twice the Laplacian standard deviation. We call this empirically derived probability C and describe it by the integral:

$$1 - \left[\int_{\frac{-2\sqrt{2}}{\alpha}}^{\frac{2\sqrt{2}}{\alpha}} \left(A\frac{\alpha}{2}e^{-\alpha|x|} + \frac{B}{\sqrt{2\pi}\sigma_{gaus}}e^{\frac{-1}{\sigma_{gaus}^2}x^2} \right) dx \right] = C.$$ (3.10)

Our choice of σ_{gaus}^2 is not a closed-form finding; we graphically determined it to be 900. Empirically , this variance may seem high when viewing the graph in Figure 3.17, but we choose to err in this direction because we want to place as much density in the outskirts of the Laplacian as reasonably possible when we calculate the quanta. These higher-valued detail coefficients carry the most impact on overall image quality, and it makes sense to give them as many quanta that can be justified by the statistics. Solving the system of equations, Equations 3.9 and 3.10, for all detail coefficients yields $A = 0.9678$ and $B = 0.0322$.

Combining the results from this section, we are able to assume that the Laplacian/Gaussian mixture pdf for the detail coefficients of the test image is

$$f_x(x) = 0.19356e^{-0.4|x|} + 0.0004282e^{\frac{-x^2}{1800}}.$$ (3.11)

Figure 3.18 shows the normalized histogram of the combined detail coefficients for all scales and images plotted with a normalized version of this pdf.
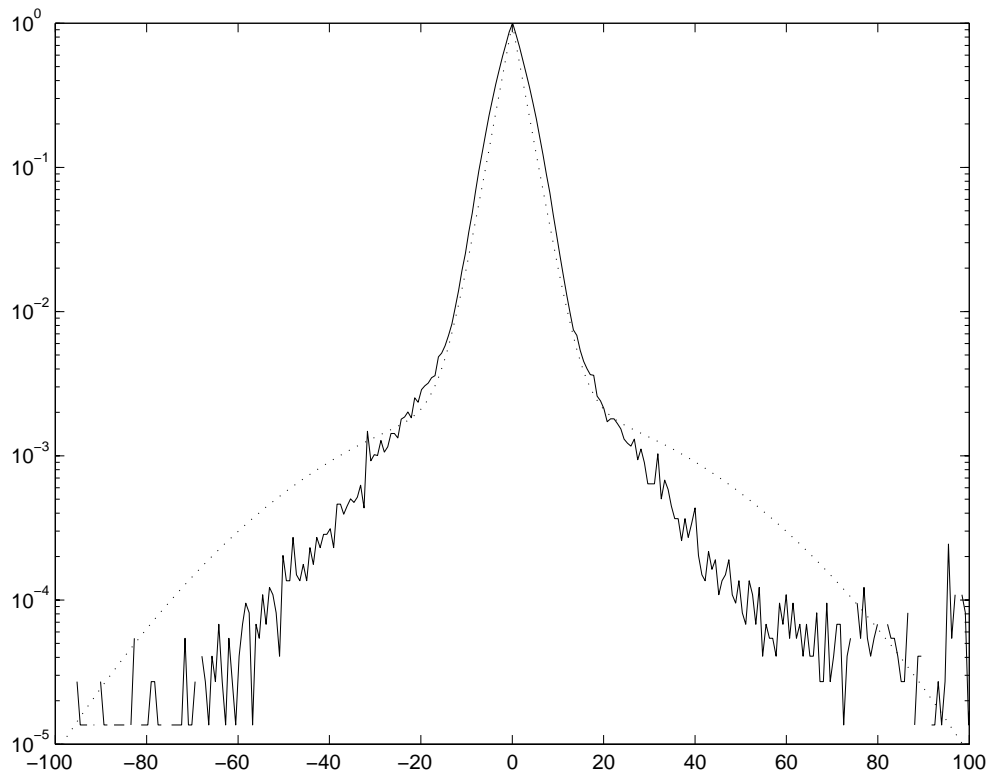
Figure 3.18.   *This logarithmic plot shows the normalized histogram of the detail coefficients in line-form along with the normalized joint Laplacian/Gaussian representation. As seen from this plot, the joint pdf better represents the histogram mass on the outskirts of the Laplacian portion.*

3.5.1 The Lloyd-Max Quanta.   As described in Section 2.3, the Lloyd-Max quantization scheme finds the optimal quantization values that minimize mean-square error based on the pdf of the data [10] [11]. The implementation of this scheme with the pdf described in Equation 3.11 yields quantization values found in Table 3.1. A more complete list of Lloyd-Max quanta is found in Appendix A. The transition levels between adjacent quanta is defined as the average of the two adjacent quanta [10] [11].

Number of Levels	Positive Quantization Values
2	3.17
4	2.6, 36.49
6	1.67, 7.86, 40.02
8	1.49, 6.51, 23.89, 50.04
10	1.14, 4.42, 10.33, 28.75, 53.12
12	1, 3.72, 8.01, 17.85, 34.88, 56.99
14	0.85, 3.04, 6.15, 11.45, 23.83, 39.45, 59.84
16	0.75, 2.62, 5.11, 8.82, 15.86, 28.23, 42.8, 61.89
18	0.67, 2.3, 4.39, 7.27, 11.88, 20.91, 32.51, 46.09, 63.86
20	0.6, 2.04, 3.82, 6.14, 9.48, 15.17, 24.66, 35.61, 48.46, 65.26

Table 3.1.   *This is a list of positive quantization levels that were calculated using the Lloyd-Max method.*

3.6   Performing the Thresholding Operation

For our algorithm, we code the image using a threshold to determine insignificant coefficients. For quantization purposes we use a soft-threshold. In order to not significantly change the quality of the image, the quantized values are adjusted during image reconstruction to their hard-threshold state. With a soft-threshold, we set all coefficients with an absolute value less than the threshold value γ to zero, subtract γ from all positive values, and add it to the negative values. This type of transformation was shown in Figure 2.5. On the histogram, this transformation is represented by setting all the histogram density that is within +/- γ on the x-axis to zero and by shifting the two remaining sides together. This section discusses how we implement the threshold operation and what the implications of thresholding are on the histogram. Since we have optimized our quantization levels according to the histograms of the detail coefficients, we need reassurance that a soft-threshold will not ruin these efforts.

Because we have removed the normalization step on the transform, we must threshold each scale of detail coefficients differently. If this were a valid wavelet transform (if we had kept the normalization step), then we would threshold each scale with the same value. However, by removing the normalization, the coefficients

are given a weight of one half (relative to the coefficients in a valid DWT), which is compounded with each scale. We therefore threshold the values at each scale with a threshold that is one half the value of the threshold used at the previous scale.

3.6.1 The Effect of a Soft-Threshold on the Optimality of our Quantization Levels.   In Section 3.5, we assumed a certain pdf for the detail coefficients of our transform. Since the optimal quantization levels are determined based on this pdf, we recognize that it is not advisable to blindly change the histogram of the detail coefficients via a soft-threshold without understanding the impact. In this section, we show that the actual pdf is not significantly changed by a soft-threshold.

We begin our justification by noting that in our application of lossy image compression, detail coefficients with a value of zero are ignored. Therefore, all co-efficients that are set to zero are removed from the resulting histogram. Also, note that the pdf is almost entirely Laplacian. If we, for the moment, assume that the pdf is *entirely* Laplacian, then a hard threshold on the pdf yields

$$
t_x(x) = \begin{cases} \frac{\alpha}{2}e^{-\alpha x} & x > \gamma \\ \frac{\alpha}{2}e^{\alpha x} & x \leq -\gamma \\ 0 & -\gamma < x \leq \gamma \end{cases}
\tag{3.12}
$$

To make a soft threshold, we want to create a transformation that shifts all the values towards zero by the value of γ. Specifically, let us set

$$
\begin{aligned}
y = x - \gamma & \quad \text{for } x > 0 \\
y = x + \gamma & \quad \text{for } x < 0
\end{aligned}
\tag{3.13}
$$

After ignoring all zero-coefficients, we combine Equations 3.12 and 3.13 to obtain

$$
f_y(y) = \frac{\alpha}{2}e^{-\alpha \gamma}e^{-\alpha|y|}.
\tag{3.14}
$$

3-29

Normalizing $f_y(y)$ to make a valid pdf reveals the original Laplacian pdf. Hence, if the pdf is entirely Laplacian, the mean-square quantization error is not changed by a soft-threshold; regardless of the threshold, the resulting pdf of the non-zero coefficients is the same.

Of course, the pdf that describes the detail coefficients in the test image is not entirely Laplacian; it contains a very minor Gaussian component as well. Therefore, additional analysis is needed.

The standard deviation of the Laplacian portion of the pdf in Equation 3.11 is $\frac{\sqrt{2}}{\alpha} \approx 3.53$, while the standard deviation of the Gaussian portion is 30. If we assume a threshold value that does not completely annihilate the Laplacian portion of the pdf, then, since the spread of the Gaussian is so much larger, the result of the threshold on the Gaussian is approximately the same Gaussian. The resulting pdf after thresholding resembles a Laplacian/Gaussian mixture pdf, where the Gaussian portion is more pronounced than before thresholding. Since the Gaussian portion of the pdf changes minimally and since the Laplacian contribution to the pdf is nearly zero on the skirts, the Lloyd-Max quanta do not change significantly over the primarily Gaussian portion. In short, the values of the quanta from the un-thresholded coefficients are a good approximation to the optimal quanta after thresholding.

3.7   Coding the Image

So far we have discussed our image enhancement techniques (FPN correction and histogram stretching), we have chosen the transform to be used, we have analyzed the statistics of the transform coefficients to determine the optimal Lloyd-Max quantization, and we have explained how we threshold each scale of detail coefficients. This section describes how we code the image to achieve compression. We initiate the discussion by describing the order in which we threshold and quantize the detail coefficients. We then discuss our method of run-length coding, which is where most of our compression benefits are realized. Once we write to disk the values and

locations of the significant detail coefficients (via run-length coding), we finish by quantizing and writing to disk the coarse image coefficients from the fifth and final iteration of the transform.

3.7.1 Organizing the Detail Coefficients. As stated previously, the four quadrants created by one iteration of the multiscale transform are similar in function to those from one iteration of the 2-dimensional DWT. The upper-left quadrant, called the LL quadrant, contains the lower resolution image. The upper-right quadrant, HL, contains low frequency information in the vertical direction and high frequency information in the horizontal direction. As a result, the information tends to be grouped together vertically. The lower-left quadrant, LH, contains high frequency information in the vertical direction and low frequency information in the horizontal direction and tends to have information grouped horizontally. The lower-right quadrant is the HH quadrant and contains high frequency information in both directions, so information is grouped together diagonally. This type of information is useful for justifying the order with which we choose to threshold and quantize the detail coefficients for each scale.

For each iteration of the transform, we threshold and quantize each detail coefficient separately. If the detail coefficient has a magnitude less than the threshold value, we count it as insignificant. We are interested in storing only the quantized significant coefficients to disk, so we choose to use a run-length coding scheme to store their locations. In essence, we store to disk the quantized value of the significant coefficients as well as the number of insignificant coefficients between them. We keep track of the number of insignificant coefficients using a consecutive zero counter.

The order in which we quantize the coefficients is shown in Figure 3.19. We start at the upper left data element in the HL quadrant and read downward, since the data in this quadrant is grouped vertically. At the end of the first column in HL, we start at the top of the second column. Once all of the coefficients in HL are

quantized, we proceed to the upper left data element in LH without resetting our consecutive zero counter. We then read the coefficients to the right since the data is grouped horizontally in this quadrant. As we read along each row, we do not stop at the end of the row in LH; we continue reading along the row of HH. We read the HH quadrant horizontally instead of diagonally because it is time-consuming to organize the coefficients diagonally. At the end of the HH row, we refocus on the first element of the next row and repeat. Once all of the detail coefficients are quantized and coded at this scale, we iterate the transform on the LL quadrant, or the coarse coefficients, and continue the process on the detail coefficients at the next scale without resetting the consecutive zero counter.
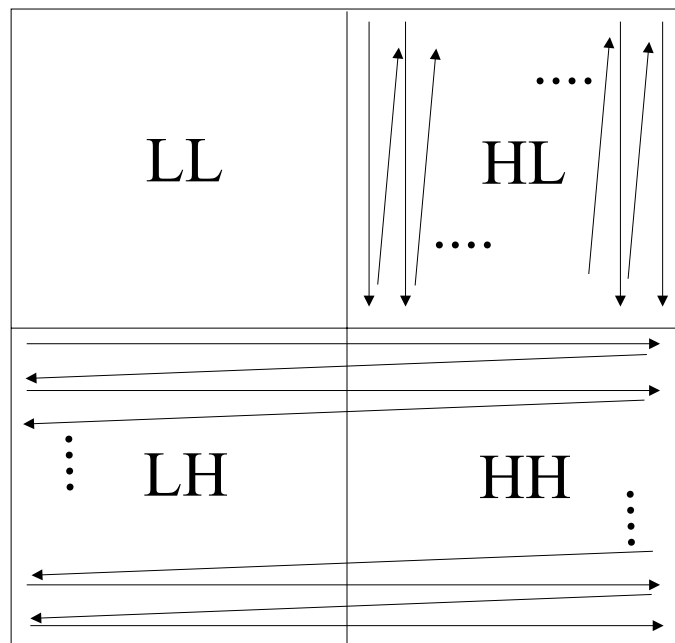
Figure 3.19.   *This diagram shows how we read the coefficients. We start at the upper left element of the HL quadrant and read down the columns. We then move to the LH quadrant and read across the rows.*

3.7.2   Implementing the Run-Length Code: Counting the Consecutive Zeros.
In run-length coding, we count the number of zeros between two significant coefficients in order to keep track of their locations. Since the quantized values of the

significant coefficients are designated by unique entropy codewords, we must find a way to write the number of consecutive zeros to disk that will not interfere with the Huffman codes. In order to accomplish this task, we propose to represent the number of consecutive zeros with a combination of elements that also have entropy codewords. To this end, we create a finite number of quantization levels called zero-count levels. We define the values of these levels so that *any* positive integer can be represented by some potentially repeated combination.

3.7.2.1   Designing the Zero-Count Levels.   The goal is to designate a finite number of levels that each represent a different number so that *any* positive integer can be represented by a combination of these different zero-count levels. An obvious solution is to have each zero-count level represent an increasing power of two. For example, let us use only eight zero-count levels for this kind of integer representation. These levels are described in Table 3.2

Zero Count Level	Number of Zeros	Integer Number of Zeros
1	2^0	1
2	2^1	2
3	2^2	4
4	2^3	8
5	2^4	16
6	2^5	32
7	2^6	64
8	2^7	128

Table 3.2.   *The all zero-count levels used in our algorithm. Any positive integer can be represented by a combination of these levels. Some of the levels can be repeated.*

As stated previously, any positive integer can be represented by a combination of these levels. For example, four levels are used to describe the number 23:

$$23 = 2^0 + 2^1 + 2^2 + 2^4$$
$$23 = 1 + 2 + 4 + 16$$

(3.15)

The number 196 is represented by three levels:

$$196 = 2^2 + 2^6 + 2^7$$
$$196 = 4 + 64 + 128$$

$$(3.16)$$

3.7.2.2   Implementing the Integer Decomposition into a Combination of Zero-Count Levels.   There are several ways to discover which combination of levels is needed to represent a given integer. One easy way is to work with the binary equivalent of the integer.

For example, suppose we want to decompose the number 302 into a series of these levels. Consider first the binary equivalent of the integer

$$302 = 100101110. \qquad (3.17)$$

Split the binary number into two sections: the seven least significant bits, and the remaining bits. The seven least significant bits, 0101110, are used to determine which of the first seven zero-count levels are used to describe the integer. In this case, the 2^1, 2^2, 2^3, and 2^5 levels are used. Next, the bits that are more significant than the least six are isolated to describe how many times the last zero-count level is used to describe the integer. In this case, the decimal-equivalent of the bits 10 is 2. The 2^7 zero-count level is used two times. The decomposition of the integer 302 is

$$302 = 2^1 + 2^2 + 2^3 + 2^5 + 2^7 + 2^7$$
$$302 = 2 + 4 + 8 + 32 + 128 + 128$$

$$(3.18)$$

3.7.3   Choosing an Entropy Code Design.   As stated in Section 2.4, the Huffman code creates an entropy code scheme for the data set based on the probability mass function (pmf) of the quantized transform coefficients. So far in this chapter, we analyzed the transform statistics of our test image once it is FPN-corrected and

histogram-stretched to determine the pdf of the detail coefficients. We discussed how we implement the soft-threshold without specifically quantifying a threshold value. Then we described how to represent the number of consecutive zeros between significant coefficients as a combination of zero-count levels. Our determination of a Huffman code is based on how many quanta we choose to use for our Lloyd-Max quantizer, how many zero-count levels we choose to use, and what value we choose for the soft-threshold. It is important to realize that although the threshold value does not significantly change the pdf of the detail coefficients (as described in Section 3.6.1), the threshold value *will* significantly affect the distribution of the zero-count levels, thereby affecting the outcome of the entropy coding.

Our goal is to calculate a single Huffman code that can be used for all our images. However, implementing a Huffman code requires prior knowledge of the probability mass function (pmf) of the quantized data. In Section 3.5, we assume a single pdf to represent the un-quantized detail coefficients at every scale for every image. Hence, we also assume that the resulting pmf of the quantized coefficients is the same. As stated in the last paragraph, the factor that would make the pmf's different, assuming the same number of quanta and zero-count levels, is the threshold value. Increasing the threshold value creates more insignificant coefficients, thereby increasing the probability of the zero-count levels. Therefore, to increase compression performance, a Huffman code must be calculated for every threshold value. If the threshold value is unknown to the decoder, then it must be included in the overhead of the coded image file. Once the decoder knows the threshold value used during the coding process, then it can determine which Huffman code was used. For our specific application, we will choose a configuration for the image coder that includes the number of quanta, the number of zero-count levels, and the threshold value, so that the same Huffman codes are used at all times. We choose to do this in the interest of processing speed. Our decision is based on the performance charts in Appendix B. These charts were made using eight zero-count levels.

We assumed a single Huffman Code appropriate for all our images. We now verify this assumption. We do this by simply applying the Huffman code calculated for one image to another image and comparing the difference in bits per level (bpl) from that of its own Huffman code. We expect the bpl value to not change significantly. For this test, we apply the Huffman codes calculated for test image 1 using 40 quanta, 8 zero-count levels, and a threshold value of 20. We then apply this code to test image 2 and 3 and see how much the bpl values change from the application of their own Huffman codes. Table 3.3 shows the results of this test. Note that the change in bpl is minimal, meaning that the entropy code for one image closely approximates the entropy code of another.

Test Image	bpl From Own Code	bpl From Image 1 Code	Percent Change
2	4.64	4.67	0.65%
3	4.45	4.49	0.90%

Table 3.3. *Huffman Code Performance Comparison Between Images. This table shows the performance of the Huffman code calculated for test image 1 as it is applied to test image 2. Note that the change in bits per level (bpl) is minimal, meaning that the entropy code for one image closely approximates the entropy code of another.*

3.7.4 Coding the Final Iteration Coarse Coefficients. After the run-length coding operation on all five iterations of detail coefficients, we round the coarse coefficients to the nearest integer and write them to disk by columns starting at the upper left data element. We designed the transform so that the range of data in the coarse image is consistent across all scales, so each element in the coarse image is rounded to the nearest 8 bit integer. Since after five iterations the coarse image size is only 15x20, the use of a prediction encoder is not worth the compression benefits of its implementation; the coarse image requires only 300 bytes of disk space.

3.8 A New Image Quality Measure

Current industry standard image quality metrics tend to emphasize the quality of lower frequency objects in an image, but we consider impulsive information very important. Thus, we require in our application a measure that considers the quality of high frequency objects as well as the quality of larger low frequency objects. In this section we introduce a new measure that gives equal contribution to the high frequency, impulsive objects and the lower frequency objects. We begin by analyzing the standard measure, Mean Square Error (MSE), on which PSNR is based. We then recognize why this measure is not optimal for evaluating impulse quality. Next, we suggest a new measure that better reflects the type of quality we are interested in maintaining in our image.

3.8.1 Parseval's Identity applied to MSE. As stated previously, the MSE measurement of an image tends to emphasize the lower frequency details. Our justification begins by defining the error:

$$\epsilon(i,j) = I_o(i,j) - I_c(i,j). \tag{3.19}$$

Equation 2.3 can be rewritten

$$MSE(I_c, I_o) = \frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{j=1}^{M} |\epsilon(i,j)|^2. \tag{3.20}$$

We use Parseval's relation to write

$$\sum_{i=1}^{N} \sum_{j=1}^{M} |\epsilon(i,j)|^2 = \frac{1}{N \cdot M} \sum_{k=1}^{N} \sum_{l=1}^{M} |\varepsilon(k,l)|^2, \tag{3.21}$$

where $\varepsilon(k,l)$ is the discrete Fourier transform of $\epsilon(i,j)$ [9]. Since the Fourier transform is a linear operation, Equations 3.19 and 3.21 can be combined with 3.20 in the following way:

$$MSE(I_c, I_o) = \left(\frac{1}{N \cdot M}\right)^2 \sum_{k=1}^{N} \sum_{l=1}^{M} |F_o(k,l) - F_c(k,l)|^2, \qquad (3.22)$$

where F_o and F_c are the discrete Fourier transforms of I_o and I_c, respectively.

If an image tends to have more low frequency content than high frequency content, which is typical due to the energy compaction property of the Fourier transform, then one easily sees that relative changes in the low frequency content will influence the result in Equation 3.22 more than the same relative changes in the high frequency content. In other words, if the DC or zero-frequency component of an image receives a 50% change, the MSE measure would be strongly affected, but a 50% change in the highest frequency component would hardly be noticed. Thus, MSE has more fidelity to the lower frequency components than to the higher frequency components.

3.8.2 A New Frequency-Based Mean Squared Error Measure. We are interested in preserving the quality of impulse-like objects as well as low-frequency objects. Since an impulse contains infinite frequency, we suggest a measure where each frequency component in Equation 3.22 is given a more equal contribution to the overall measurement. We call this modified MSE measure the Weighted Frequency Mean Square Error (WFMSE). It takes the form

$$WFMSE(I_c, I_o) = \left(\frac{1}{N \cdot M}\right)^2 \sum_{k=1}^{N} \sum_{l=1}^{M} W(k,l) \cdot |F_o(k,l) - F_c(k,l)|^2. \qquad (3.23)$$

The question is how to design $W(k,l)$. Let us make the heuristic observation that a frequency component in the original image that changes due to compression by a factor of β is the same error in either direction. In other words, we would like to say that whether the frequency component changes by β or $\frac{1}{\beta}$, we treat the error the same. Therefore, we define

$$W(k,l) = \frac{R}{\max\left(|F_o(k,l)|^2, |F_c(k,l)|^2\right)}, \tag{3.24}$$

where R is a normalization constant. This weight assures bounds on the resulting ratio, specifically

$$0 \leq \frac{|F_o(k,l) - F_c(k,l)|}{\max\left(|F_o(k,l)|, |F_c(k,l)|\right)} \leq 2. \tag{3.25}$$

Two is the maximum value because it is possible for $F_o(k,l)$ to be equal and opposite in magnitude (or 180° out of phase) with $F_c(k,l)$.

It is easily seen that if there is no error $(F_o(k,l) = F_c(k,l))$, then $WFMSE(I_c, I_o) = 0$. It would be desirable if, in the extreme case that $F_c(k,l) = 0$, we will obtain the same result as we would with MSE. If we assume that $F_c(k,l) = 0$ for all k and l, then Equation 3.22 can be reduced to

$$MSE(0, I_o) = \frac{1}{N \cdot M} \cdot \overline{|F_o|^2}, \tag{3.26}$$

where $\overline{|F_o|^2}$ is the mean square value of F_o. The same reduction of Equation 3.23 yields

$$WFMSE(0, I_o) = \frac{1}{N \cdot M} \cdot R. \tag{3.27}$$

Forcing Equations 3.26 and 3.27 to be equal suggests that $R = \overline{|F_o|^2}$. This conclusion is apparently not very reassuring considering that the maximum of every frequency component in this case came from $F_o(k,l)$. But since we choose to make a factor change in either direction the same, the choice of normalization factor is of a relative matter. Our wish is to normalize the measure to the average square frequency component in $F_o(k,l)$ at all times, regardless of which image has the maximum

frequency-component-values. Therefore, we set $R = \overline{|F_o|^2}$ and make the following observation using Parseval's relation

$$R = \frac{1}{N \cdot M} \sum_{k=1}^{N} \sum_{l=1}^{M} |F_o(k,l)|^2 = \sum_{i=1}^{N} \sum_{j=1}^{M} |I_o(i,j)|^2. \qquad (3.28)$$

The complete form of WFMSE is created by combining Equations 3.23, 3.24, and 3.28. At this point we make some observations about this new measure. Obviously, if $F_o(k,l) = F_c(k,l)$, our new measure reflects zero error, which is consistent with the traditional MSE measure. Also, if $F_c(k,l) = 0$, then the MSE (Equation 3.26) is the same as the WFMSE. Further analysis of Equation 3.26 using Parseval's relation reveals the mean squared value (MSV) of the original image. Finally, if $F_o(k,l) = -F_c(k,l)$, then the two images are equal and opposite in magnitude. We note that both WFMSE and MSE result in a value of 4 times the MSV of the original image.

3.8.3   A Frequency Based Peak Signal to Noise Ratio Measure.   We now create a measure similar to PSNR that is based on our new measure, WFMSE. The numerator of the logarithm in Equation 2.2 is assumed to be the MSV of an image where every element is at its maximum possible value. In 8-bit imagery, the maximum possible value of each pixel element is 255. As discussed in the previous section, the mean square value of an image is the same as the mean square error of that image with an image of zeros. Thus, Equation 2.2 can be rewritten as

$$PSNR(I_c, I_o) = 10 \log_{10} \left(\frac{MSE(0, 255)}{MSE(I_c, I_o)} \right). \qquad (3.29)$$

We propose to create a measure similar to PSNR that is based on our new metric, WFMSE. This measure is created by merely replacing the MSE operator in Equation 3.29 with the WFMSE operator. This new measure, called weighted frequency peak signal to noise ratio (WFPSNR) is

$$WFPSNR(I_c, I_o) = 10 \log_{10} \left(\frac{WMSFE(0, 255)}{WMSFE(I_c, I_o)} \right), \qquad (3.30)$$

where, naturally, $WMSFE(0, 255) = 255^2$.

3.9 Conclusion

In this chapter, we presented the methodology of our research. The goals are to compress an image with fixed pattern noise as quickly as possible while maintaining impulsive features in the image. The chapter begins with an analysis of the image. Due to application constraints, the camera must be run in a mode that creates a dominant noise pattern across the image array. We observed that this noise pattern is fixed and repeatable on the array and characterizable. Therefore, we must remove this pattern from the image because it clearly degrades the viewable information in the image. We must also remove the pattern before compression because lossy compression will change the character of the noise, disrupting our ability to remove the noise pattern after compression. After the removal of the noise pattern, we stretched and clipped the histogram to maximize compression performance. This histogram stretch does not cost anything in terms of processing speed because it can be implemented at the same time the fixed pattern noise correction is implemented.

Next we presented the lifting implementation of the wavelet transform as it applies to our routine. During this discussion, we chose the Cohen-Daubucies-Feauveau 3,1 biorthogonal wavelet transform based on computational speed. We also justified the removal of the normalization factor from the lift for the sake of computational simplicity. This removal invalidates the transform as a wavelet transform, but many of the desirable properties of the wavelet transform are still preserved.

We then analyzed the statistics of the transform detail coefficients in order to determine the Lloyd-Max quanta, which minimize quantization error in the mean square. A brief discussion followed that described how a soft-threshold on the detail

coefficients minimally impacts the optimality of the Lloyd-Max quantization values due to the fact that the detail coefficients of the transform tend to follow a Laplacian/Gaussian mixture distribution.

Next we discussed specifics of how we code the image. Because of thresholding, many of the coefficients were determined to be insignificant and were designated zero. In order to gain compression, we use a run-length coding scheme to store significant coefficient locations. The number of consecutive zeros between each significant coefficient is decomposed into a combination of numbers that are assigned codewords.

Finally, we introduced new measures of image quality specifically designed to evaluate the quality of impulsive features in the image. We showed that current image quality measures, like PSNR or MSE, tend to emphasize the quality of the low frequency content in an image. Since impulsive features contain infinite frequency, it makes more sense to use a measure that attempts to give equal contribution to all frequency components. Our new measures, weighted-frequency mean squared error (WFMSE) and weighted-frequency peak signal to noise ratio (WFPSNR) were created with this need in mind.

IV. Results

4.1 Introduction

In this chapter, we report the performance of the image coder outlined in Chapter 3. We begin by showing that the compression routine produces a compressed image in which objects are visually discernable, which we demonstrate using three different compression levels. Next, we compare the JPEG standard to our image coder when it is configured for 40 quantization levels and 8 zero count levels. This configuration of our image coder is called Q40. Our comparison is accomplished by plotting the PSNR and WFPSNR performance for both coders versus compression ratio (CR) for test image 1. We also compare the images associated with these plots, showing that the quality of the impulses suffer with JPEG but not with Q40. Next, we show the JPEG images at the compression ratio where impulse quality begins to become unacceptable versus that for the Q40 images. Finally, we discuss algorithm speed, comparing the speed of JPEG compression versus the compression of our Q40 image coder.

4.2 Experimental Results

In this section we demonstrate that the images produced by our 40-quanta, 8-zero-count image coder, Q40, is of good visual quality. Our first demonstration is shown in Figure 4.1. In this figure, Test Image 1 is compressed by a ratio of 10. Note that all objects in the scene are easily discernable and that there are no degrading artifacts overwhelming the image. The same can be shown for Test Image 2, which is compressed by a ratio of 20, as shown in Figure 4.2. In Figure 4.3, we show that Q40 compresses Test Image 3 by a ratio of 40 with no overwhelming compression artifacts.

Figure 4.1.   *Test Image 1: Compression Ratio = 10. Here, test image 1 was compressed using our 40-quanta image coder by a ratio of 10.*
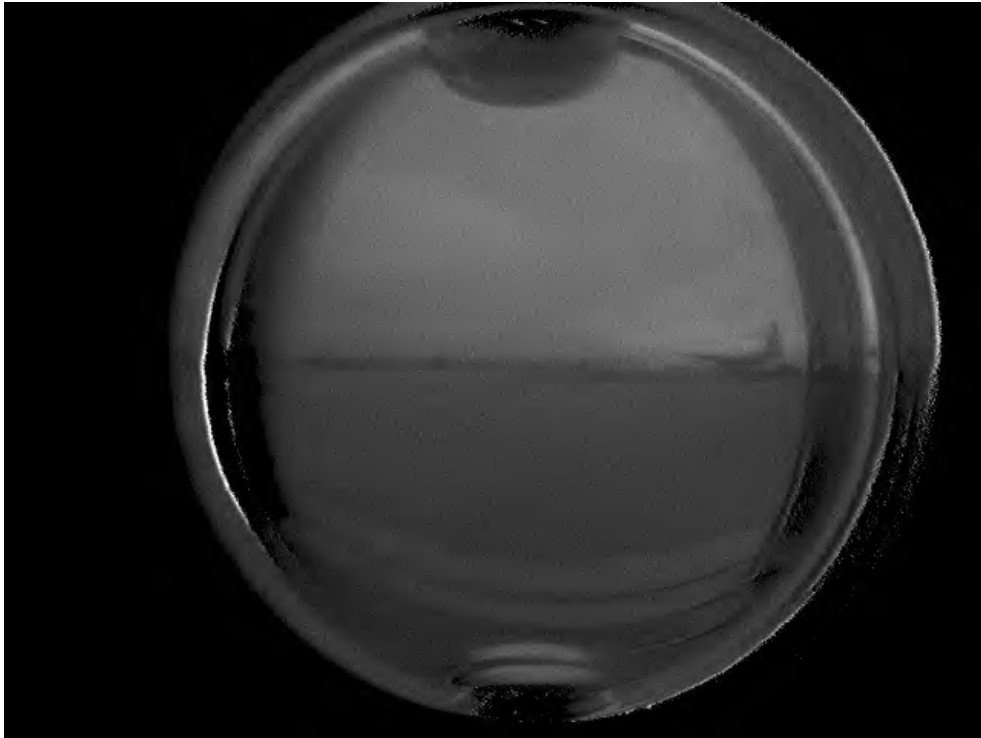
Figure 4.2.　*Test Image 2: Compression Ratio = 20. Here, test image 2 was compressed using our 40-quanta image coder by a ratio of 20.*
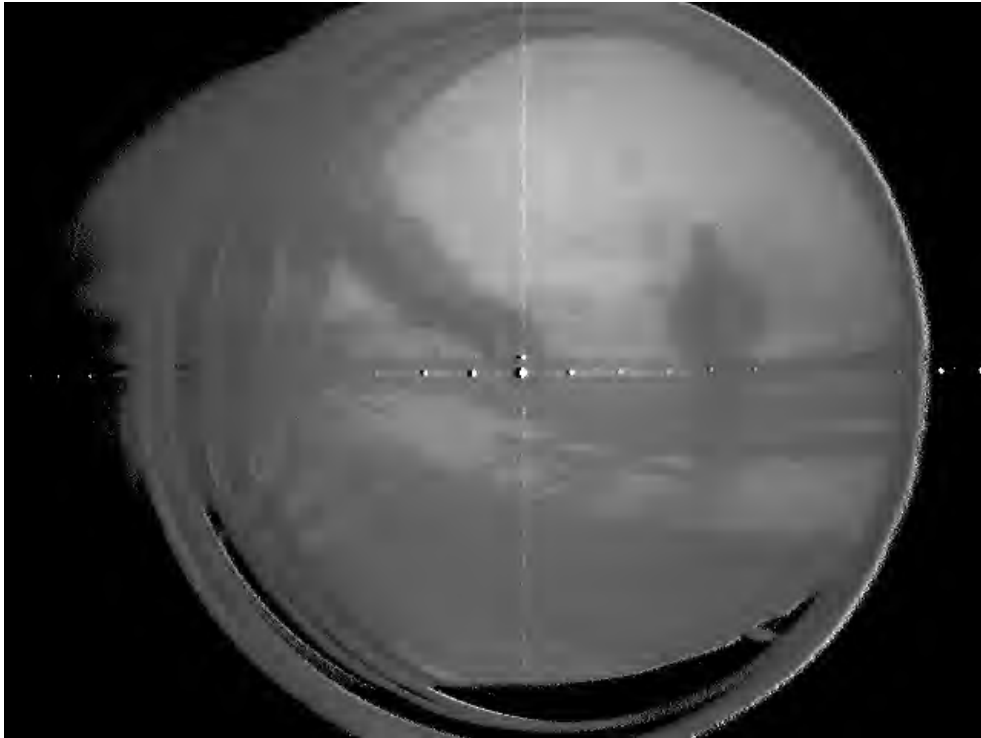
Figure 4.3.   *Test Image 3: Compression Ratio = 40. Here, test image 3 was compressed using our 40-quanta image coder by a ratio of 40.*

4.3   Compression Performance of Q40 Against JPEG

The plots in Figure 4.4 show the performance curves for test image 1 of our Q40 image coder against the performance curves for JPEG in terms of PSNR versus compression ratio in (a) and WFPSNR versus compression ratio in (b). Note that Q40 barely outperforms JPEG in PSNR for lower compression ratios, but clearly outperforms JPEG in WFPSNR. A clip of test Image 1 at the operational points designated by gray circles in Figure 4.4 (b) is shown in Figure 4.5. These images clearly show that our Q40 image dominates JPEG in preserving the quality of the impulsive features. In (a) JPEG is seen to have obliterated the impulses entirely, while they remain completely intact in (b).

4.3.1   Preserving Impulses in Compression.   The image examples in Figure 4.5 show that JPEG eliminates the impulses when compressing Test Image 1 to a CR of 40, while Q40 preserves them. In this section, we show the limitations of these two compression routines based entirely on their ability to preserve these impulses. Figure 4.6 shows Test Image 1 with a JPEG compression ratio of 15. With just a compression ratio of 15, it is easy to visually discriminate objects in the image. However, we see compression artifacts that change the physical shape of the pulses through induced ringing. The ringing, which is shown in the image strip (b), suggests a bound on the JPEG compression ratio due to impulse quality. The Q40 image coder can achieve much higher levels of compression before the quality of the impulses begins to suffer, as shown in Figure 4.7. In this image, we increased the threshold value to the point where the threshold begins to negatively affect the quality of the impulses. From the image clip in Figure 4.7 (b), we see that one of the pulses is entirely obliterated from the threshold operation (marked by the white circle). The other pulses, however, are still unaffected. While this image is unacceptable in terms of visual quality, it shows that we constrain our Q40 compression on bounds dictated solely by our ability to visually discriminate objects
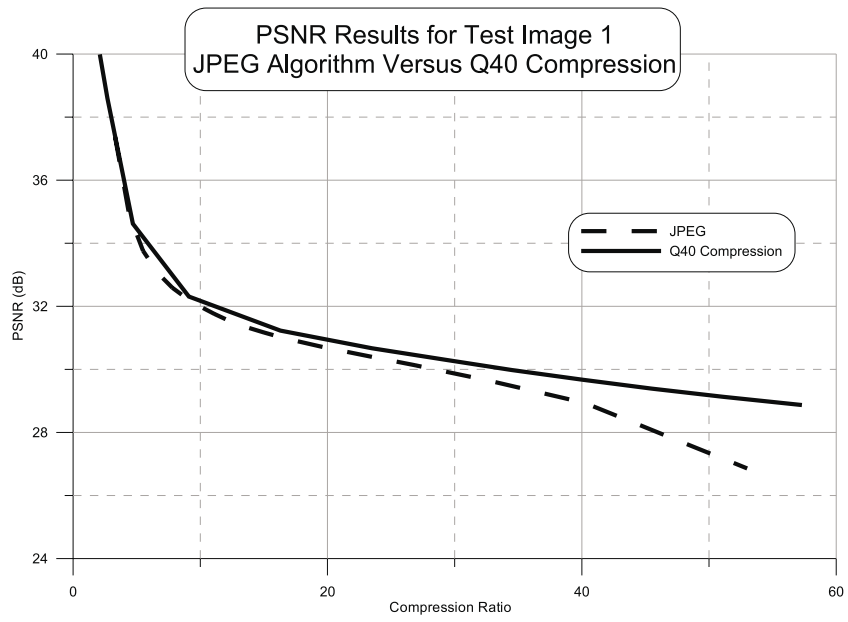
(a)
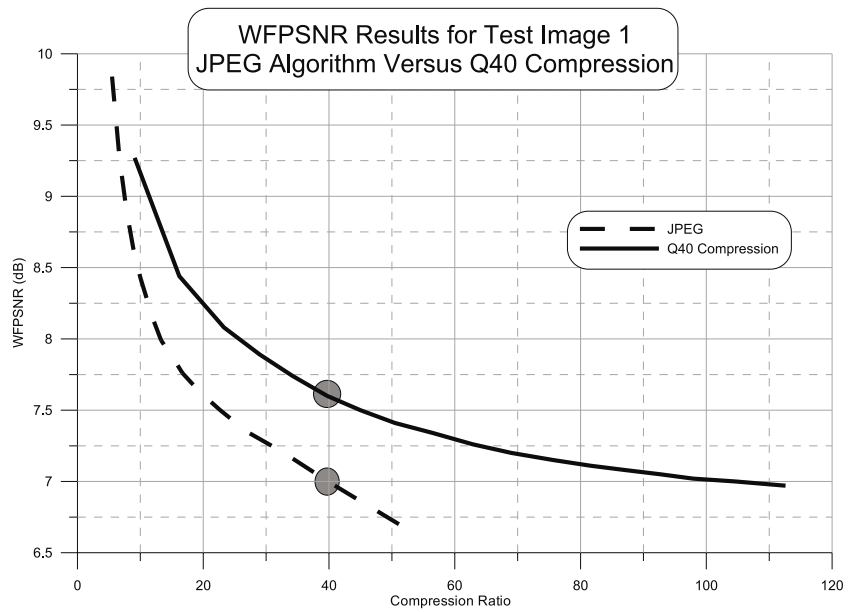
(b)

Figure 4.4.   *These plots show the compression performance in terms of (a) PSNR and (b) WFPSNR of our 40-quanta image coder (Q40) and JPEG. Note that in (a), Q40 code barely outperforms JPEG in PSNR, but in (b), the Q40 coder clearly outperforms JPEG in WFPSNR. The gray circles in (b) show the operational points on the curve for which we compare the images in Figure 4.5.*

4-6

(a)	(b)
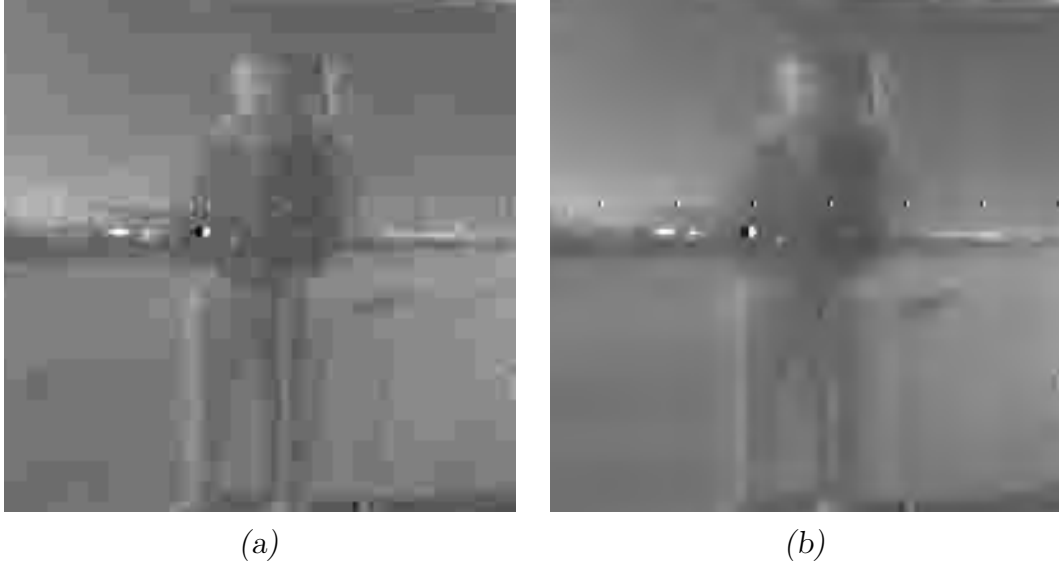
Figure 4.5.　　*Test Image 1: JPEG Versus Q40 at a Compression Ratio of 40. (a)*
　　　　　　　Test Image 1 is compressed using JPEG to a compression ratio of 40
　　　　　　　and a WFPSNR of 6.99 dB. (b) Test Image 1 is compressed using
　　　　　　　Q40 to a compression ratio of 40 and a WFPSNR of 7.6 dB. Note that
　　　　　　　JPEG clearly eliminates the impulsive features in the image while Q40
　　　　　　　clearly preserves them.

in the image. Conversely, we constrain the JPEG compression on bounds dictated
solely by the quality of impulses in the image.

4.4　Q40 Implementation Time versus JPEG Implementation Time

In this section we compare the implementation time of the Q40 image coder
to that of JPEG. We implement the Q40 coder in C++ using custom functions.
The JPEG coder is implemented in C++ using special JPEG libraries found on the
Internet [7]. Table 4.1 shows the average time it takes to implement the different
compression components of Test Image 1 using our C++ implementation of the Q40
image coder. Table 4.2 shows the implementation time of the JPEG compression on
the same image using the same computer at the maximum acceptable compression
ratio of 15. Although the Q40 coder is slightly slower than JPEG, our C++ code

(a)
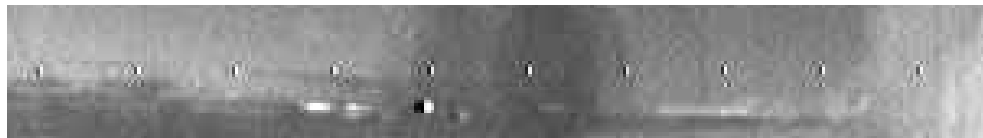
(b)

Figure 4.6.   *Test Image 1: JPEG compression by a ratio of 15. These images sug-gest a bound on JPEG compression ratio for test image 1. While the objects in this image are visually discernable (a), the quality of the im-pulses begins to suffer (b). Note that the ringing around each impulse in (b) produces ambiguity in actual impulse location and shape.*

(a)

(b)

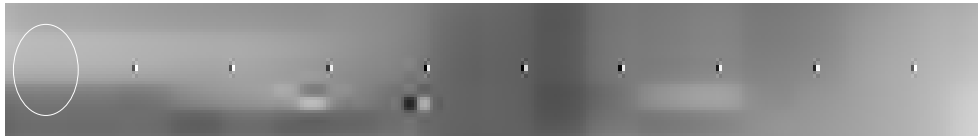Figure 4.7.   *Test Image 1: Q40 compression by a ratio of 190. These images suggest a bound on Q40 compression ratio for test image 1. Object discrimination in the image (a) is nearly ruined. Impulse discrimination (b), however, remains nearly in tact. As shown by the white circle, one of the impulses is eliminated due to the high compression.*

for the Q40 image coder is not fully optimized for speed. Further optimization may make the code run as fast or faster than JPEG.

Process	Time (ms)
FPN Correction	60.7
Transform	37.7
Quantization, Coding, and Writing	17.7
Total	**116.1**

Table 4.1.   *The average time required to implement the different compression components of Test Image 1 using the Q40 image coder. Compared to JPEG, this implementation is slower, but the C++ code may be further optimized.*

Process	Time (ms)
FPN Correction	60.7
JPEG Compression and Writing	30.2
Total	**90.9**

Table 4.2.   *The average time required to implement the different compression components of Test Image 1 using JPEG. Compared to the unoptimized implementation of Q40, JPEG is faster.*

V. Discussion and Future Work

5.1 Contributions of this Thesis

In this thesis, we designed a transform image coder that is very fast and faithful to the visual and impulse quality of the image. We began by characterizing and removing a dominant fixed pattern noise from the image. Next we used a histogram-stretch to improve the visual quality of the image, thus making compression more effective. We then used an image transform based on the Cohen-Daubechies-Feauveau 3,1 biorthogonal wavelet transform implemented with a reverse lift. We adjusted the normalization step so that each scale of the transform occupied the same relative dynamic range. The result was a transform that is very fast and computationally simple. Our discussion then turned to an analysis of the transform statistics for three test images, on which we based the design of the Lloyd-Max quanta. We then showed how we coded the image very quickly using a run-length method.

After designing the image coder, we looked for ways to accurately quantify its performance. The three main measures of interest included implementation speed, bit rate, and image quality. Quantifying image quality is relative. Standard quality measures like MSE and PSNR seem to be good measures for visual quality comparisons, but they tend to emphasize low frequency objects in the image. In our application, impulsive features are very important, so we designed a measure that gives more weight to impulse quality than MSE and PSNR.

Finally we compared our new image coder with JPEG based on the measures discussed above. We demonstrated that our image coder outperforms JPEG in image quality versus bit rate. We also demonstrated that our image coder is very fast at forward image compression and rivals JPEG in compression speed.

5.2   Recommendations for Future Work

5.2.1   Optimization of the image coder.   The image coder described above performs well with regard to bit rate and compression ratio. The speed comparison, however, is based entirely on the optimization of the image coder. We may be able to further optimize the code to increase algorithm speed.

5.2.2   Post Compression FPN correction.   Currently, we remove the fixed pattern noise before we compress the image because the lossy compression changes the character of the fixed pattern noise, degrading our ability to remove it based on its pre-compression characterization. The removal of this pattern, while effective in improving visual quality, takes considerable time to implement. It is not evident that the compression will change the characterization of the noise in an unpredictable way. Therefore, it might be worthwhile to characterize the fixed pattern noise after image compression in an attempt to remove the pattern after decompression, when there is no time constraint. This sort of study would involve a different threshold method, redesigned quanta, and a quantization error analysis.

5.2.3   Comparison to Other Techniques.   In this thesis, we compared one configuration of our image coder to JPEG. It would be desirable to compare our image coder to other wavelet-based image coders, such as the Shapiro embedded zero tree algorithm [15] or Said and Pearlman's set partitioning method [14]. These algorithms would have to be redesigned for our images and optimized for speed.

Appendix A.   Table of Lloyd-Max Quanta

Tables A.1 and A.2 in this appendix contain the Lloyd-Max quanta calculated for the probability density function in Equation 3.11. The transition levels between adjacent quanta is defined by Lloyd as the average of the two adjacent quanta.

2	4	6	8	10	12	14	16	18	20
3.17	2.60	1.67	1.49	1.14	1.00	0.85	0.75	0.67	0.60
X	36.49	7.86	6.51	4.42	3.72	3.04	2.62	2.30	2.04
X	X	40.02	23.89	10.33	8.01	6.15	5.11	4.39	3.82
X	X	X	50.04	28.75	17.85	11.45	8.82	7.27	6.14
X	X	X	X	53.12	34.88	23.83	15.86	11.88	9.48
X	X	X	X	X	56.99	39.45	28.23	20.91	15.17
X	X	X	X	X	X	59.84	42.80	32.51	24.66
X	X	X	X	X	X	X	61.89	46.09	35.61
X	X	X	X	X	X	X	X	63.86	48.46
X	X	X	X	X	X	X	X	X	65.26

Table A.1.   *This table lists the Lloyd-Max quanta calculated from the pdf in Equation 3.11. The row across the top of the table is the number of quantization values. The quanta listed are positive values only; since the pdf is symmetric about zero, there are also corresponding negative values.*

22	24	26	28	30	32	34	36	38	40
0.55	0.51	0.47	0.43	0.41	0.38	0.36	0.34	0.32	0.31
1.84	1.67	1.54	1.42	1.32	1.23	1.16	1.09	1.03	0.98
3.40	3.06	2.78	2.55	2.36	2.19	2.05	1.92	1.81	1.71
5.37	4.75	4.28	3.88	3.56	3.29	3.06	2.85	2.68	2.52
8.02	6.93	6.13	5.49	4.99	4.57	4.22	3.91	3.65	3.43
12.04	9.96	8.58	7.54	6.75	6.11	5.59	5.15	4.78	4.46
19.02	14.76	12.13	10.31	9.03	8.03	7.26	6.62	6.09	5.65
28.23	22.32	17.79	14.45	12.20	10.58	9.38	8.44	7.68	7.06
38.57	31.12	25.36	20.66	16.97	14.22	12.26	10.79	9.67	8.78
50.73	40.97	33.75	28.00	23.33	19.45	16.37	14.04	12.30	10.97
66.56	52.56	43.15	36.04	30.39	25.72	21.82	18.54	15.92	13.90
X	67.60	54.22	45.04	38.10	32.52	27.91	23.98	20.64	17.85
X	X	68.51	55.66	46.76	39.94	34.47	29.88	25.99	22.63
X	X	X	69.30	56.95	48.29	41.63	36.22	31.70	27.83
X	X	X	X	69.99	58.10	49.69	43.15	37.84	33.37
X	X	X	X	X	70.59	59.15	50.95	44.55	39.32
X	X	X	X	X	X	71.13	60.09	52.11	45.84
X	X	X	X	X	X	X	71.61	60.96	53.17
X	X	X	X	X	X	X	X	72.05	61.75
X	X	X	X	X	X	X	X	X	72.44

Table A.2. *This table lists the Lloyd-Max quanta calculated from the pdf in Equation 3.11. The row across the top of the table is the number of quantization values. The quanta listed are positive values only; since the pdf is symmetric about zero, there are also corresponding negative values.*

Appendix B.   Operational Performance Contour Plots

In this appendix, we report how the operational performance of the image coder changes as a function of number of quanta and threshold value for each test image. These plots are helpful in determining where we should operate for our application. In these plots we use 8 zero-count levels

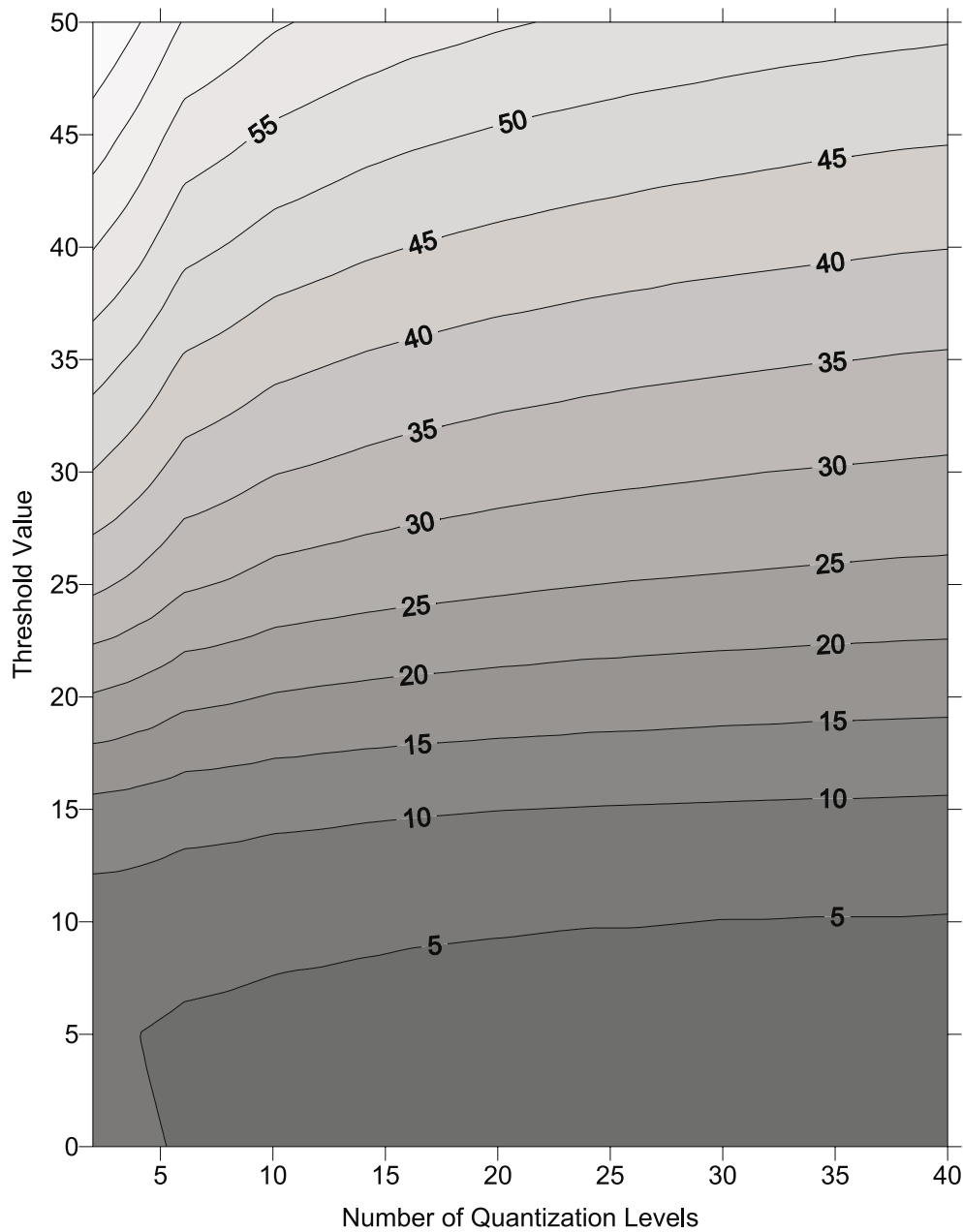# Test Image 1: Compression Ratio

Figure B.1.   *Test Image 1: Operational Contour Plot of Compression Ratio. This contour plot shows the lines of constant compression ratio as the number of quanta and the threshold value vary. Note that at low threshold levels, compression ratio changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*
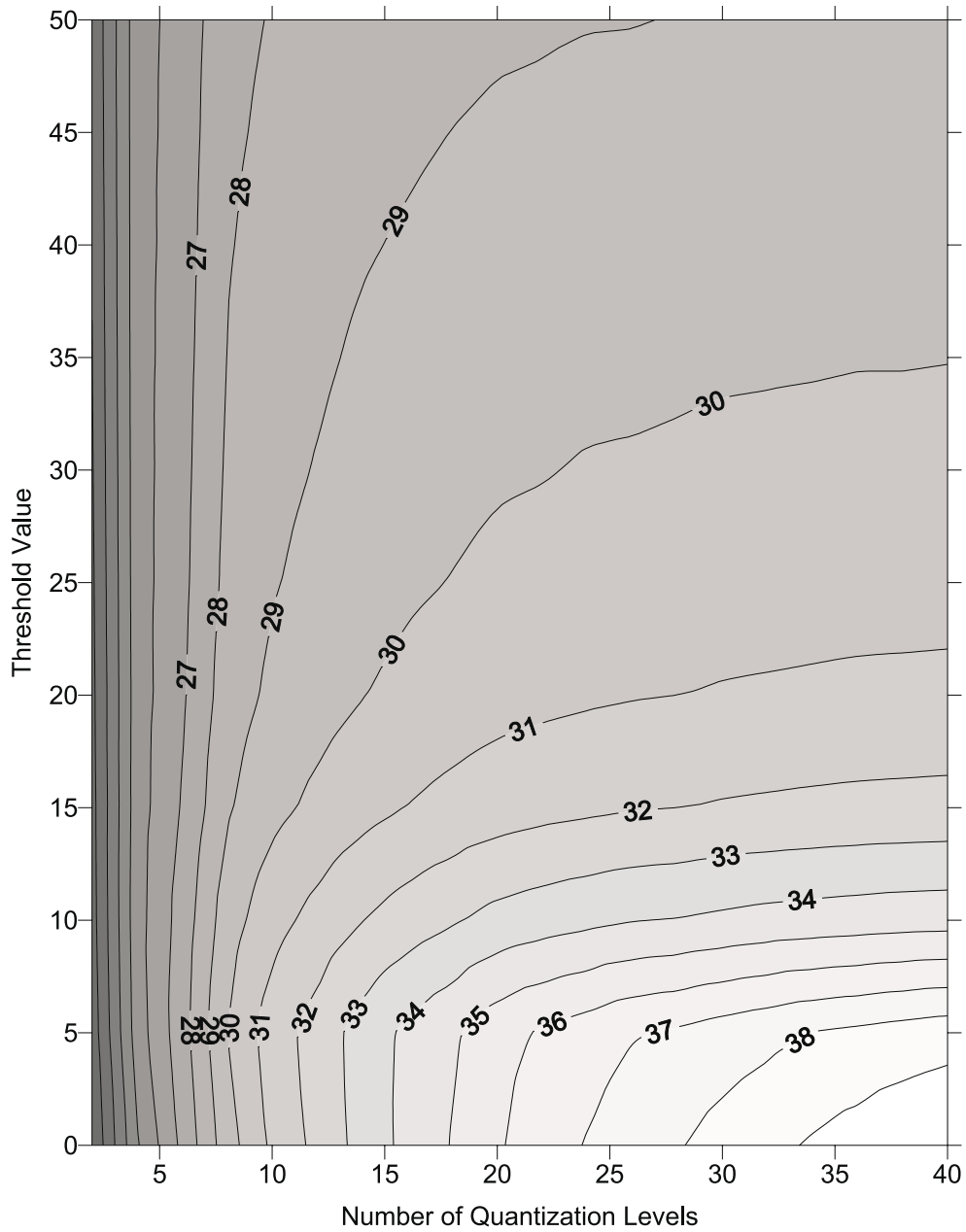
Figure B.2.   *Test Image 1: Operational Contour Plot of PSNR. This contour plot shows the lines of constant PSNR as the number of quanta and the threshold value vary. Note that at low threshold levels, PSNR changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*
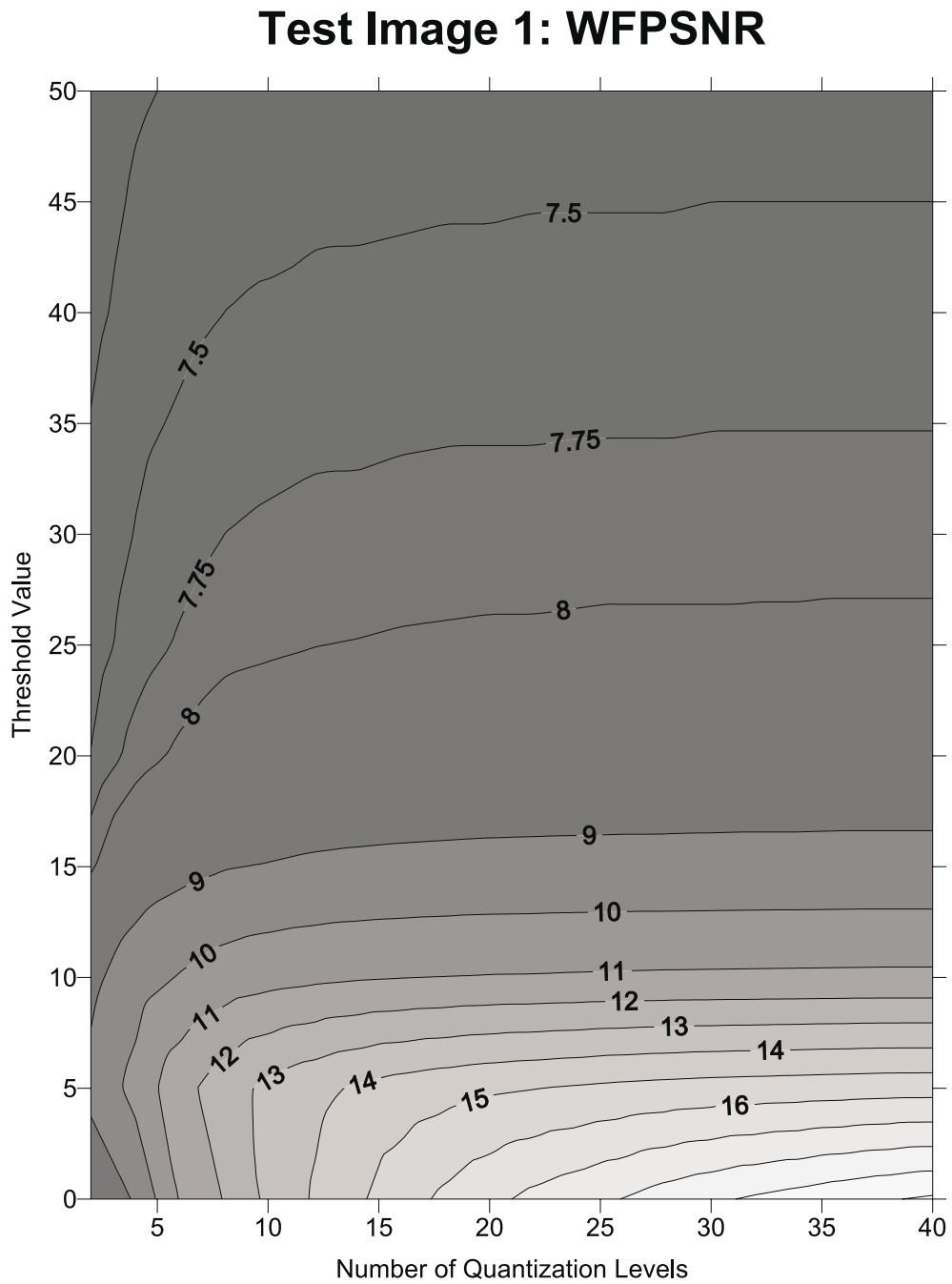
Test Image 1: WFPSNR

Figure B.3.   *Test Image 1: Operational Contour Plot of WFPSNR. This contour plot shows the lines of constant WFPSNR as the number of quanta and the threshold value vary. Note that at low threshold levels, WFPSNR changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*
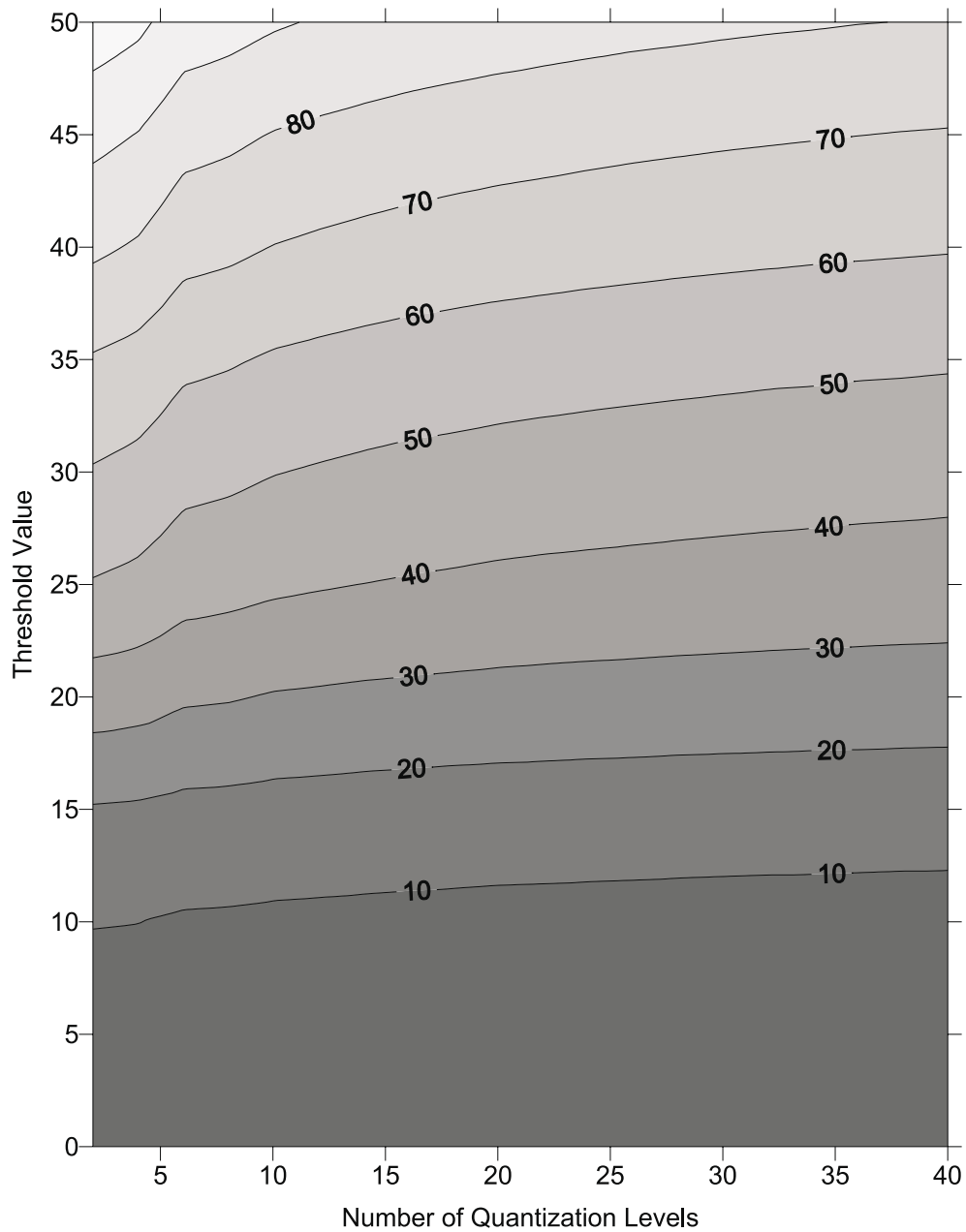
Figure B.4.   *Test Image 2: Operational Contour Plot of Compression Ratio. This contour plot shows the lines of constant compression ratio as the number of quanta and the threshold value vary. Note that at low threshold levels, compression ratio changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*

Test Image 2: PSNR

Figure B.5. *Test Image 2: Operational Contour Plot of PSNR. This contour plot shows the lines of constant PSNR as the number of quanta and the threshold value vary. Note that at low threshold levels, PSNR changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*

Test Image 2: WFPSNR

Figure B.6.   *Test Image 2: Operational Contour Plot of WFPSNR. This contour plot shows the lines of constant WFPSNR as the number of quanta and the threshold value vary. Note that at low threshold levels, WFPSNR changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*

Test Image 3: Compression Ratio

Figure B.7. *Test Image 3: Operational Contour Plot of Compression Ratio. This contour plot shows the lines of constant compression ratio as the number of quanta and the threshold value vary. Note that at low threshold levels, compression ratio changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*
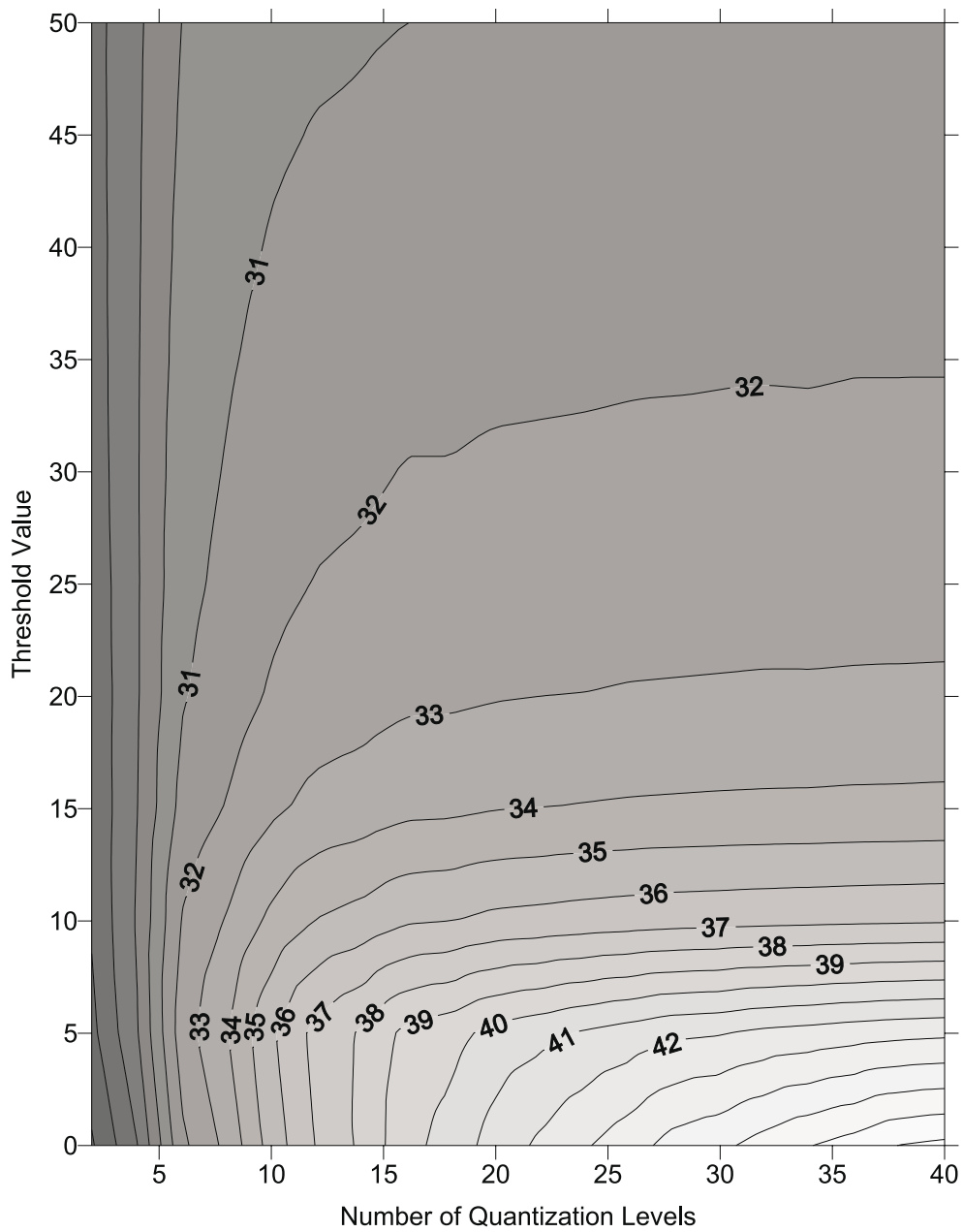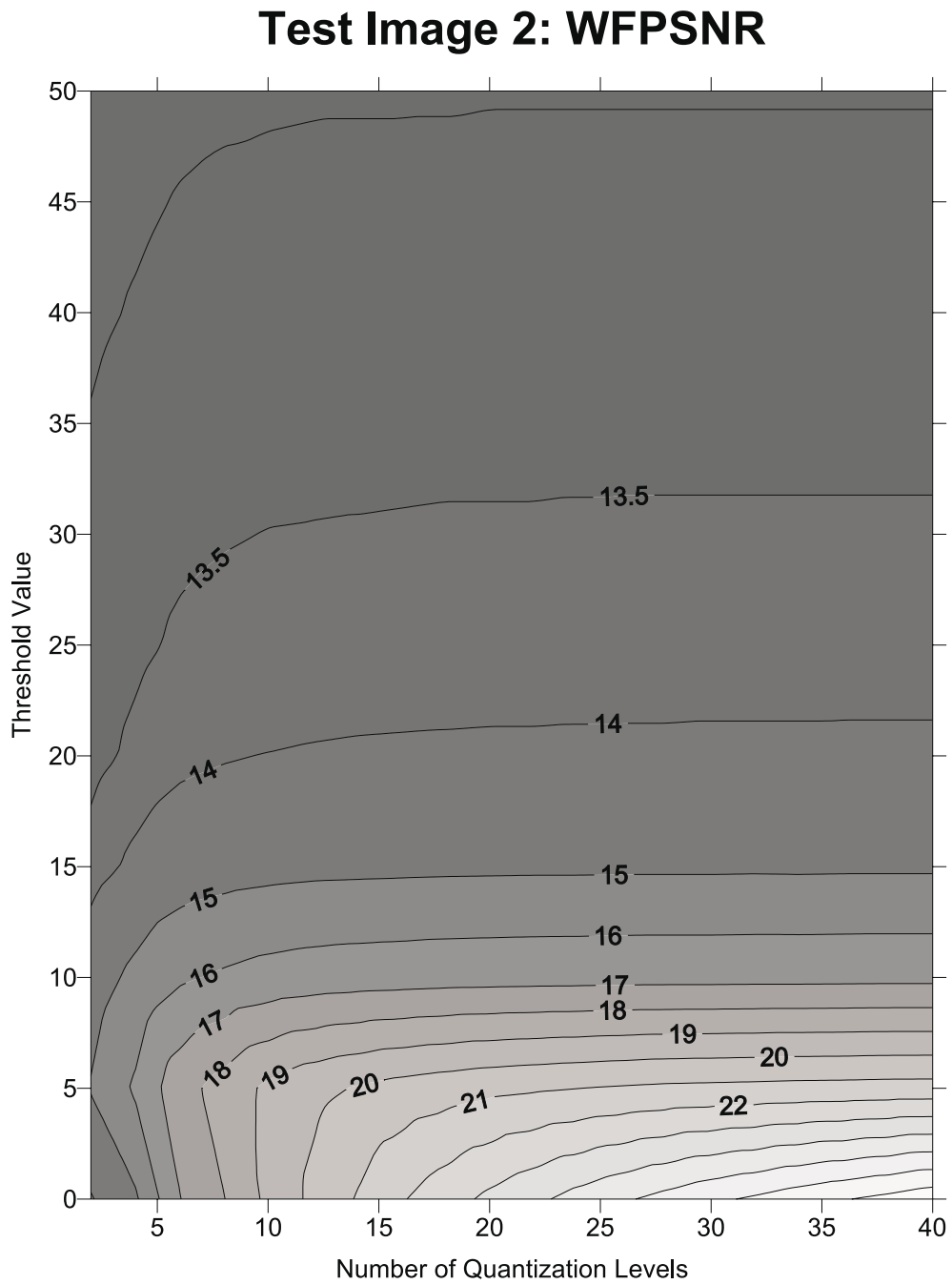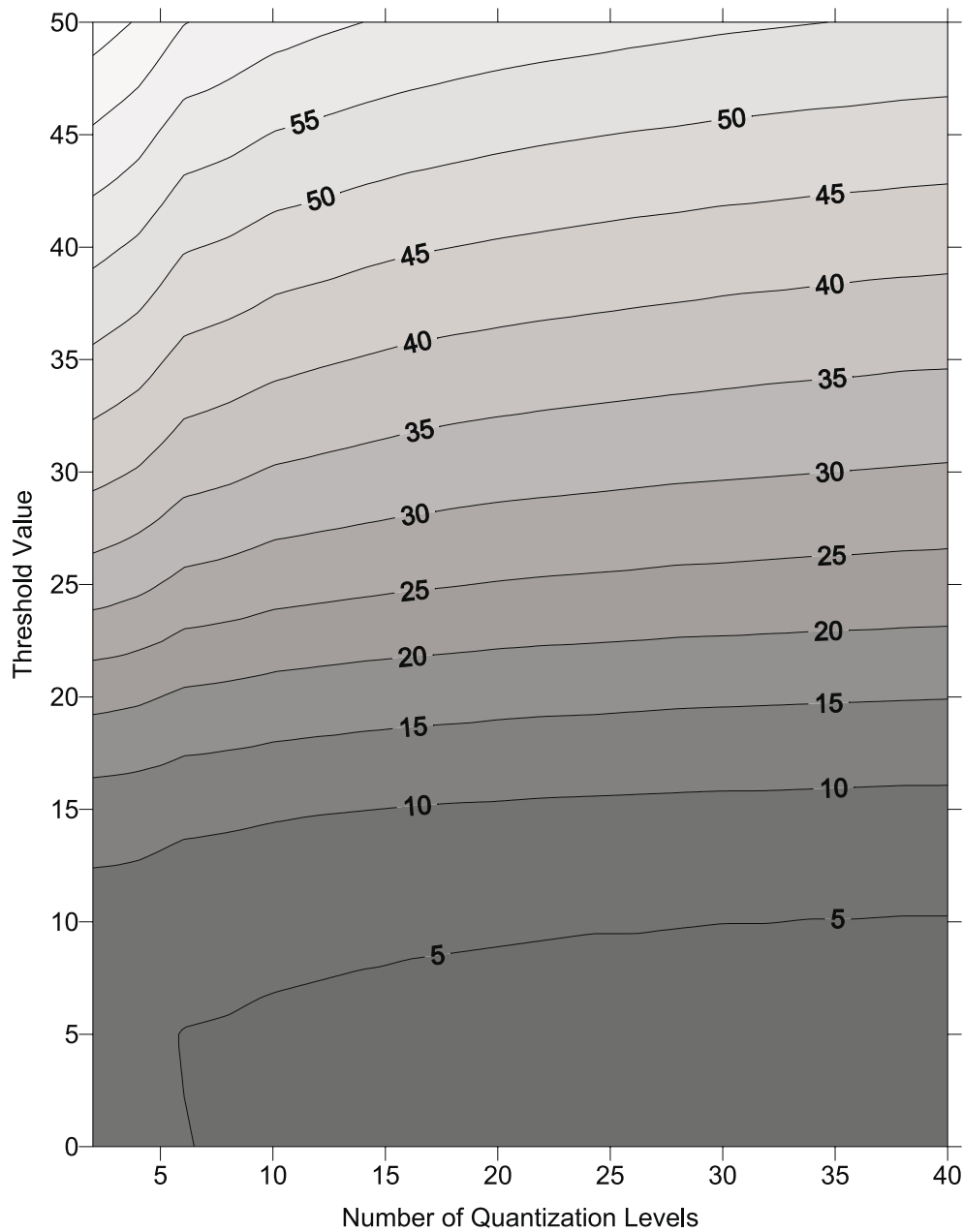
Test Image 3: PSNR

Figure B.8.   *Test Image 3: Operational Contour Plot of PSNR. This contour plot shows the lines of constant PSNR as the number of quanta and the threshold value vary. Note that at low threshold levels, PSNR changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*
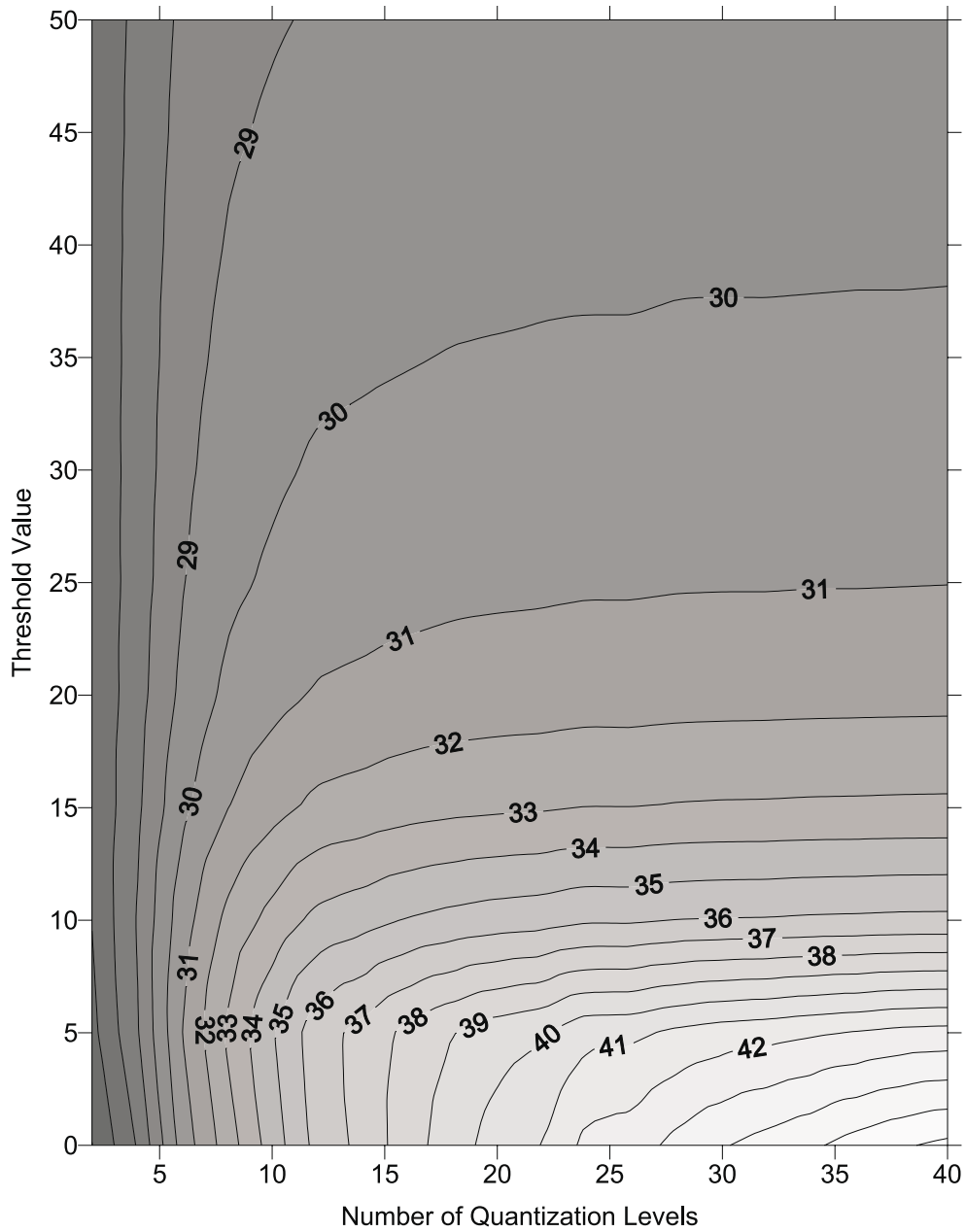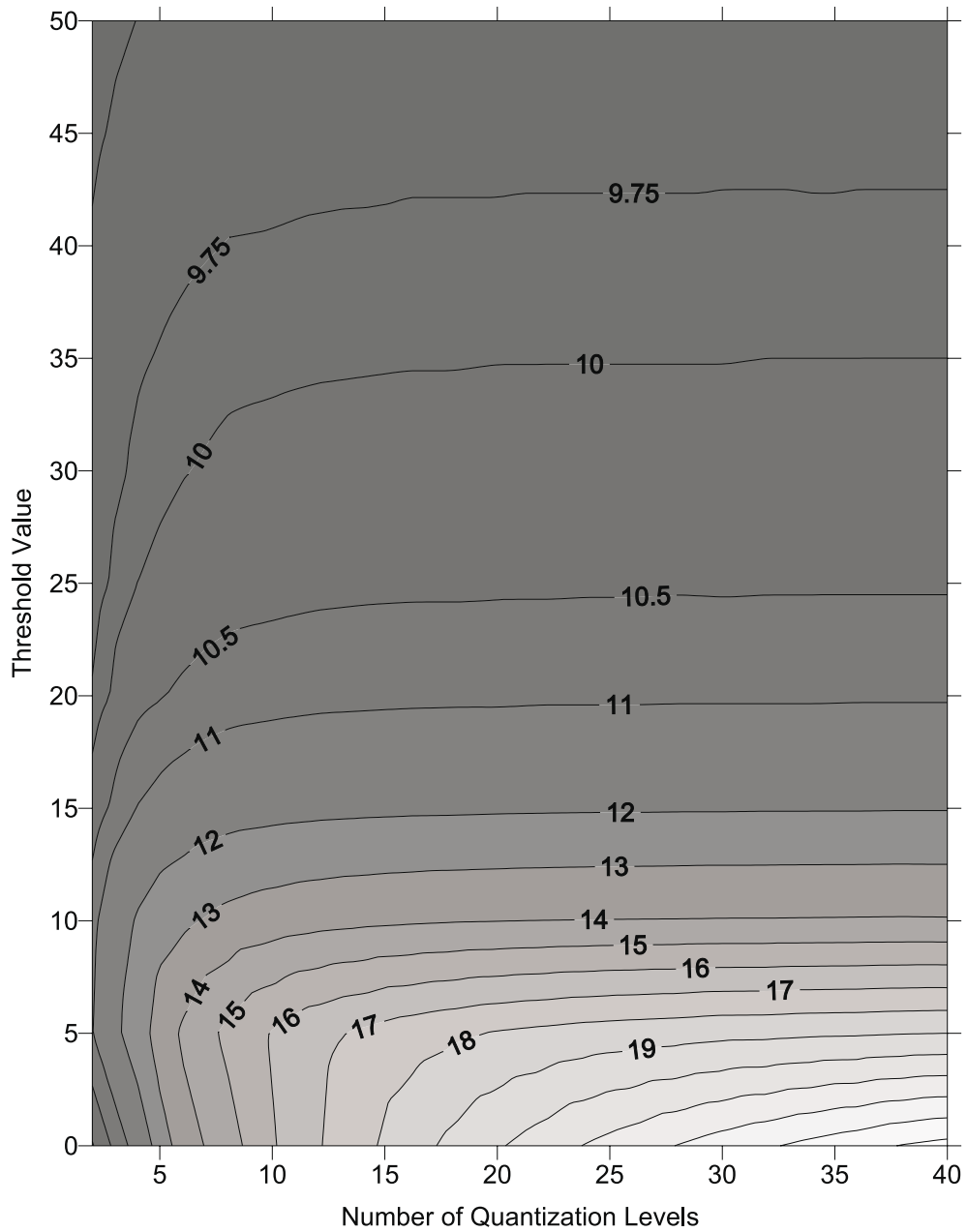
Figure B.9.   *Test Image 3: Operational Contour Plot of WFPSNR. This contour plot shows the lines of constant WFPSNR as the number of quanta and the threshold value vary. Note that at low threshold levels, WFPSNR changes significantly as the number of quanta increases. The same is not as true at higher threshold values.*

Bibliography

1. Burrus, C. S., et al. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

2. Claypoole, R. L. and R. Baraniuk "Flexible Wavelet Transforms Using Lifting," *Porceedings of the 68th Annual Meetign of the Society of Exploration Geophysicists*, New Orleans, September 1998.

3. Daubechies, I. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conf. Series in Appl. Math., Vol. 61, Philadelphia, PA: SIAM, 1992.

4. Donoho, D. "De-noising by Soft-Thresholding," *IEEE Trans. Inform. Theory*, *41*(3):613-627 (1995).

5. Holst, G. C. *CCD Arrays Cameras and Displays*. Winter Park, FL: JDC Publishing, 1996.

6. Huffman, D. A. "A Method for the Construction of Minimum-Redundancy Codes," *Procedings of the I.R.E.*, *8*(2):1098-1101 (1951).

7. Independent JPEG Group, "Independent JPEG Group" http://www.ijg.org/.

8. Leon-Garcia, A. *Probability and Random Processes for Electrical Engineers* (2nd Edition). Reading, MA: Addison-Wesley, 1994.

9. Lim, J. S. *Two-Dimensional Signal and Image Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1990.

10. Lloyd, S. P. "Least Square Quantization in PCM," *IEEE Trans. Inform. Theory*, IT-28 :129-137 (1982).

11. Max, J. "Quantizing for minimum distortion," *IEEE Trans. Inform. Theory*, IT-6 :7-12 (1960).

12. Morimoto, C. and R. Chellappa "Electronic Digital Image Stabilazation and Mosaicking" in *Visual Information Representation, Communication, and Image Processing*. Ed. Change Wen Chen and Ya-Qin Zhang. New York: Marcel Dekker, Inc., 1999.

13. Reichel, J., et al "Integer Wavelet Transformation for Embedded Lossy to Lossless Image Compression," *IEEE Trans. Image Proc.*, *10*(3):383-392 (2001).

14. Said, A. and W. Pearlman "A New, Fast, and Efficient Image Codec Based on Set Partioning in Hierarchical Trees," *IEEE Trans. Ciruits and Sys. for Video Tech.*, *6*(3):243-250 (1996).

15. Shapiro, J. "Embedded Image Coding Using Zerotrees of Wavelet Coefficients," *IEEE Trans. Signal Proc.*, *41*(12):3445-3462 (1993).

16. Sweldens, W. "The Lifting Scheme: A Custon-Design Construction of Biorthogonal Wavelets," *J. Appl. Comp. Harm. Anal.*, *3*(2):186-200 (1996).

17. Vetterli, M. and J. Kovačević *Wavelets and Subband Coding*. Upper Saddle River, NJ: Prentic Hall P T R, 1995.

Vita

D. Scott Anderson received his bachelor degree in Electrical Engineering at the University of Dayton in 1999. His current position is as a systems engineer at the Laser Sensor Technology Laboratory at Wright-Patterson Air Force Base, Dayton, Ohio.

REPORT DOCUMENTATION PAGE				*Form Approved* *OMB No. 074-0188*

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 25-03-2003	2. REPORT TYPE **Master's Thesis**	3. DATES COVERED (From – To) Oct 2001 – Mar 2003

4. TITLE AND SUBTITLE FAST COMPRESSION OF IMAGERY WITH HIGH FREQUENCY CONTENT	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Anderson, D. Scott	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way, Building 640 WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GE/ENG/03-21

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/SNJT BLDG 620 Attn: Dr. Jack Parker 2241 Avionics Circle, STE. 2 WPAFB OH 45433-7304	DSN: 785-9272 e-mail: Jack.Parker@wpafb.af.mil	10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Image compression is an active research area due to the many applications involving electronic media. Much research has been focused on image quality versus bit rate and/or algorithm speed. Here, we seek an effective image coder with a weighted constraint on speed. However, the compression must not taint the quality of impulsive features in the image. Moreover, the camera is operated in a mode that creates a dominant fixed pattern noise across the image array, degrading visual quality and disrupting compression performance. We propose a method that efficiently compresses such an image. We begin by characterizing and removing the fixed pattern noise from the image, thereby dramatically improving its visual quality. We follow noise removal with a histogram, or contrast, stretch. We then choose a transform that can be implemented rapidly with basic arithmetic operations and suggest a fast way to code the transform coefficients.

Image quality is a great concern. We are particularly interested in preserving pixel-sized impulsive features in the image. We seek a method of quantifying image quality based not only on the high energy low-frequency objects but also on the smaller energy, high-frequency impulsive objects. Standard image quality measures, such as mean squared error (MSE), tend to emphasize the quality of high energy objects and give less weight to the quality of pixel-sized impulses. Therefore, we develop a new measure that gives high-frequency impulsive features a greater contribution than MSE to the overall quality.

15. SUBJECT TERMS
Image Compression, Image Processing, Noise Reduction, Wavelet Transforms, Quality

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Roger L. Claypoole, Maj, USAF (ENG)
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	93	19b. TELEPHONE NUMBER (Include area code) (937) 255-3636, ext 4620; e-mail: Roger.Claypoole@afit.edu
U	U	U			

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18